دانشکده مهندسی کامپیوتر

پیاده‌سازی سیستم شناسایی و تشخیص چهره در فیلم همراه با یادگیری برخط

پایان‌نامه برای دریافت درجه کارشناسی

در رشته مهندسی کامپیوتر

محمد محمدی

استاد راهنما:

محمد رضا محمدی

مهر ماه ۱۴۰۱


بسم الله الرحمن الرحيم

چکیده

شبکه‌های عصبی در زمینه‌های متنوعی مانند پردازش تصویر، پردازش متن، بازی، پزشکی، اقتصادی و ... کاربرد دارند. با توجه به توسعه شبکه‌های عصبی و افزایش دقت آنها و بسترهای رسانه و ساخت و اشتراک‌گذاری رسانه، استفاده از شبکه‌های عصبی در کاربردهای علم داده و پردازش ویدیو و تصاویر و استخراج اطلاعات از داده مورد توجه قرار گرفته است. در واقع شبکه‌های عصبی می‌توانند به جای انسان قرار بگیرند و کارهای او از جمله شغل‌های نظارتی و تحلیل داده را به جای او و حتی بهتر از او با کیفیت و سرعتی که برای انسان مقدور نیست انجام دهند.

آنالیز و تحلیل داستان سریال‌های تلویزیونی و فیلم‌ها اصولا به شناخت شخصیت‌ها و اینکه آن‌ها در فیلم مشغول انجام چه کاری هستند نیازمند است. با پیشرفت مدل‌های آشکارساز چهره مبتنی بر شبکه‌های عصبی عمیق، حالا این امر به نظر امری قابل انجام می‌آید. گرچه که با پیشرفت مدل‌های آشکارساز چهره، دسته‌بندها و سیستم‌های تشخیص چهره نیازمند بررسی مجدد هستند و باید با توجه به گوناگونی‌های فراوانی که در سال‌های اخیر به ظاهر چهره‌ها افزوده شده است و پیشرفت‌هایی که مدل‌های آشکارساز چهره داشته‌اند تغییراتی کنند.

در این پژوهش ما تلاش می‌کنیم که شخصیت‌های افراد حاضر در ویدیو را بر اساس تصاویر چهره‌ی آن‌ها بصورت خودنظاره‌گر دسته‌بندی کنیم. تاکید اصلی ما بر روی تقطیر ضروری‌ترین اطلاعات، یعنی هویت شخص، از بازنمایی‌های بدست آمده با استفاده از شبکه‌های عمیق تشخیص چهره از پیش آموزش دیده است. ما در این پژوهش یک روش بر مبنای شبکه‌ی Siamese خودنظاره‌گر را پیشنهاد می‌کنیم که بدون نیاز به نظارت و داده‌ی برچسب خورده و بصورت برخط آموزش داده می‌شود.

برای آموزش و ارزیابی سیستم و الگوریتم‌های پیشنهادی یک پایگاه داده شامل ۲۴ دقیقه ویدیو جهت آموزش برخط سیستم و ۱۰ دقیقه ویدیوی برچسب خورده جهت ارزیابی سیستم شامل تصویر ۱۲۹۷۴ تصویر چهره جمع آوری شده است و برچسب‌های هر چهره توسط یک عامل انسانی مشخص شده‌است. ارزیابی الگوریتم نشان می‌دهد که تشخیص شخصیت هر چهره با دقت بیش از ۹۷.۶۷ ٪ بدرستی تخمین زده شده است.

واژه‌های کلیدی: تشخیص چهره، تحلیل ویدیو، هوش مصنوعی، شبکه‌های عصبی، شبکه‌های همگشتی عمیق، تشخیص کاراکترها در فیلم، یادگیری برخط، بی‌درنگ.

فهرست مطالب

عنوان صفحه

فهرست شکل‌ها

عنوان **صفحه**

فهرست جدول‌ها

عنوان **صفحه**

فهرست نمودارها

عنوان **صفحه**

فصل ۱:
مقدمه

۱-۱- شرح مسأله

امروزه با پیشرفت فناوری و سهولت دسترسی به ابزارهای مختلف، کیفیت زندگی انسان‌ها در بسیاری از حوزه‌ها با پشرفت‌های شگرفی روبرو شده است. این پیشرفت‌ها در حوزه‌های مختلف زندگی انسان‌ها را لمس کرده‌اند، چه با افزایش سهولت، چه با افزایش سرعت و از ابعاد زندگی شخصی انسان‌ها گرفته تا ابعاد صنعتی و اجتماعی زندگی انسان‌ها را با تغییرات همراه کرده است. یکی از حوزه‌هایی که اخیرا مورد توجه شدید متخصصین هوش مصنوعی بوده و پژوهش‌های زیادی را به خود اختصاص داده حوزه‌ی پردازش تصاویر ویدیویی و استخراج اطلاعات مفید از آن‌ها است. پیشرفت‌های اخیر این حوزه، ابزارهای این حوزه را به ابزارهایی قابل استفاده و قابل اعتماد در زندگی روزمره انسان‌ها تبدیل کرده است. این ابزارها کاربردهای بسیار زیادی در جامعه می‌توانند داشته باشند که از نظارت بر خطوط تولید و کنترل کیفیت کارخانه‌ها گرفته تا کمک به بررسی بهتر و دقیق‌تر تصاویر دوربین‌های امنیتی در فضاهای امنیتی می‌توانند به کار گرفته شوند. در حال حاضر بررسی تصاویر دوربین‌های امنیتی و عبور و مرور افراد مختلف از یک محل، کاری زمان‌بر است که توسط نیروی انسانی انجام می‌گیرد. این کاربرد از جمله زمینه‌هایی است که می‌توان هوش مصنوعی را جایگزین نیروی انسانی کرد و در هزینه و زمان صرفه‌جویی کرد. همچنین بررسی شخصیت‌های اصلی و فرعی فیلم‌های سینمایی و تفکیک کاراکترها و پردازش خط داستانی فیلم بر اساس کاراکترها نیز از جمله‌ی کاربردهایی است که با بکار گیری هوش مصنوعی و پردازش تصویر می‌توان به بهینگی و دقتی دست یافت که با نیروی انسانی غیر ممکن و یا بسیار پر هزینه است.

مطالعه بر روی خودکار سازی فرآیند بررسی رسانه‌های ویدیویی و استخراج اطلاعات مفید از آن‌ها با استفاده از تکنیک‌های پردازش تصویر و هوش مصنوعی دارای سابقه‌ی خوبی است و پژوهش‌های زیادی در مورد آن‌ها صورت گرفته که ما با استفاده از برخی از این کارهای پیشین و بهبود جزئی آن‌ها سعی در رسیدن به هدف خود داریم. در [۱] از روش BCL استفاده شده است که یک روش تحت نظارت است که فضای Embedding را به توپ‌هایی با سایز یکسان، یک توپ به ازای هر دسته، دسته‌بندی می‌کند و با یادگیری شعاع مناسب برای این توپ‌ها سعی در دسته‌بندی درست فضای حالت (تعداد دسته‌ها و اعضای هر دسته) می‌کند. در [۲] از یک شبکه‌ی Siamese خود-نظاره‌گر از قبل آموزش دیده استفاده می‌کند و ایده‌ی آن آموزش شبکه SSiam بدون استفاده از اطلاعات مربوط به تصاویر و دنبال کردن شخصیت‌ها است که در آن به دقت به نسبت خوبی هم رسیده‌اند.

از جمله ایرادهایی که اکثر این مطالعات دارند تحت نظارت بودن آن‌ها به هر شکلی و فرض بر ثابت گذاشتن صحنه و جایگاه دوربین در دیتاست‌ها است که ما به منظور برطرف کردن این مشکلات در این

پژوهش از تکنیک‌هایی مانند تشخیص تغییر صحنه و ردیابی اشیا سیار و تشخیص چهره نقطه به نقطه و پیاده‌سازی روش‌های خود-نظاره‌گر و آموزش آن‌ها با استفاده از اطلاعات استخراج شده از خود ویدیو بهره خواهیم برد که در نهایت با ورود تصویر شکل ۱ پس از پردازش‌های لازم خروجی شکل ۲ را داریم که تصویر پردازش‌های لازم بر روی آن صورت پذیرفته است و اشخاص حاضر در صحنه تشخیص داده شده و تصاویر چهره‌ی آن‌ها جدا شده و شناسایی شده‌اند.

شکل ۱ یک فریم از ویدیو که بعنوان ورودی دریافت می‌شود

شکل ۲ تصویر خروجی با چهره‌ی شناسایی شده‌ی اشخاص

شکل ۳ تصویر چهره‌های شناسایی شده‌ی اشخاص که با بردار ویژگی‌های آن‌ها همراه است

الگوریتم‌های مبتنی بر یادگیری عمیق در سال‌های اخیر پیشرفت‌های چشم‌گیری داشته‌اند و توانسته‌اند به دقت قابل قبول و بالایی در مسائل مختلف از جمله استخراج چهره و تشخیص چهره [۴] [۳] و دسته‌بندی تصاویر [۵] دست یابند. از مهمترین دلایل موفقیت‌های اخیر این حوزه از شبکه‌های عمیق می‌توان استفاده از معماری یادگیری سلسله مراتبی، استفاده از لایه‌های همگشتی عمیق، با در نظر گرفتن اطلاعات موجود در رابطه همسایگی میان پیکسل‌های تصویر و اشتراک‌گذاری وزن‌های شبکه، را نام برد. در این پژوهش تلاش ما طراحی الگوریتمی است که با توجه به موارد فوق‌الذکر بصورتی بهینه و با استفاده از اطلاعات موجود در خود تصاویر ویدیو بتوانیم کاراکترهای موجود در ویدیو را شناسایی کنیم و رفتار و حرکات آن‌ها را پردازش و تحلیل کنیم.

با کمک اطلاعاتی که در پایان از فریم‌های ویدیو استخراج می‌شود می‌توان اطلاعاتی همچون رفت و آمد افراد مختلف و بررسی تردد افراد و تعداد اشخاص و ساعات تردد اشخاص را در مکان‌هایی همچون بانک‌ها و ادارات با استفاده از دوبین‌های مدار بسته‌ی آن مکان‌ها را بدست آورد و همچنین اطلاعاتی همچون تعداد شخصیت‌ها و تعاملات بین شخصیت‌ها در یک فیلم سینمایی را می‌توان بدین شکل بررسی کرد و اطلاعات سودمندی را بصورت خودکار و بدون-ناظر بدست آورد.

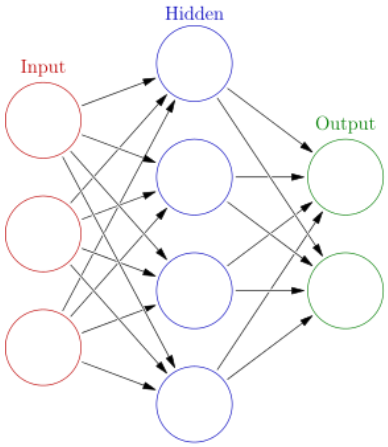
فصل ۲:
مروری بر پیشینه تحقیق

۱-۲- مقدمه

در این فصل به بررسی و طبقه‌بندی یافته‌های تحقیقات دیگر محققان می‌پردازیم. در ابتدا به تعریف برخی از اصول و مفاهیم شبکه‌های عصبی می‌پردازیم. در ادامه به چند شبکه‌ی عصبی حوزه تشخیص چهره و مسیر در تصویر و ویدیو اشاره می‌کنیم. سپس توضیحاتی درباره روند تغییرات شبکه‌های عصبی در حوزه تشخیص چهره و مسیر در تصویر و ویدیو و مراحل عمومی سامانه‌هایی که برای این منظور توسعه داده شده‌اند ارائه می‌کنیم.

۲-۲- شبکه عصبی

شبکه‌های عصبی مصنوعی (Artificial Neural Networks - ANN) یا به زبان ساده‌تر شبکه‌های عصبی، سیستم‌ها و روش‌های محاسباتی نوین برای یادگیری ماشینی، نمایش دانش و در انتها اعمال دانش به دست آمده در جهت بیش‌بینی پاسخ‌های خروجی از سامانه‌های پیچیده هستند. ایده‌ی اصلی این گونه شبکه‌ها تا حدودی الهام‌گرفته از شیوه‌ی کارکرد سیستم عصبی زیستی برای پردازش داده‌ها و اطلاعات به منظور یادگیری و ایجاد دانش می‌باشد. عنصر کلیدی این ایده، ایجاد ساختارهایی جدید برای سامانه‌ی پردازش اطلاعات است[۶] .



شکل 4 ساختار شبکه‌های عصبی

این سیستم از شمار زیادی عناصر پردازشی فوق‌العاده به هم پیوسته با نام نورون تشکیل شده که برای حل یک مسئله با هم به صورت هماهنگ عمل می‌کنند و توسط سیناپس‌ها (ارتباطات الکترومغناطیسی)

اطلاعات را منتقل می‌کنند. در این شبکه‌ها اگر یک سلول آسیب ببیند بقیه سلول‌ها می‌توانند نبود آن را جبران کرده، و نیز در بازسازی آن سهیم باشند. این شبکه‌ها قادر به یادگیری‌اند. مثلاً با اعمال سوزش به سلول‌های عصبی لامسه، سلول‌ها یاد می‌گیرند که به طرف جسم داغ نروند و با این الگوریتم سیستم می‌آموزد که خطای خود را اصلاح کند. یادگیری در این سیستم‌ها به صورت تطبیقی صورت می‌گیرد، یعنی با استفاده از مثال‌ها وزن سیناپس‌ها به گونه‌ای تغییر می‌کند که در صورت دادن ورودی‌های جدید، سیستم پاسخ درستی تولید کند. در شکل ۴ ساختار کلی شبکه‌ی عصبی را مشاهده می‌کنید.

۱-۲-۲- تعریف مجموعه‌داده

مجموعه داده به داده‌هایی گفته می‌شود که با موضوعیت و خواص مشخص و یکسان، جهت انجام تحقیقات و پروژه‌های مربوط به علم داده به جهت کسب دانش از داده‌ها استفاده می‌شوند. البته یک کاربرد دیگر مجموعه‌داده‌ها نیز برای مقایسه بین روش‌های مختلف است، به این صورت که به‌طور نمونه بر روی مجموعه داده "الف"، دو روش (الگوریتم) مختلف را اجرا کرده و با توجه به نتایج می‌توان بر اساس معیارهای دقت، سرعت و پیچیدگی هریک از روش‌ها را مقایسه کرد.

در واقع مهم‌ترین ابزار یک پژوهش‌گر برای ارائه‌ی تحلیل درست، داده‌های مورد استفاده است، و استفاده از داده‌های ناقص می‌تواند منجر به خطا در تحلیل شود و اثرات زیان‌باری در زمینه‌ی تصمیم‌گیری بر مبنای داده‌ها داشته باشد [۷] . در شکل۵ نمونه‌ای از داده‌های مورد استفاده در این پژوهش را مشاهده می‌کنید

شکل ۵ نمونه‌ای از تصاویر مجموعه‌داده مورد استفاده

۲-۲-۲- کاربرد مجموعه‌داده‌ها

استفاده از مجموعه‌داده‌ها به این صورت است که ابتدا همه یا قسمتی از داده‌های موجود در یک مجموعه‌داده را برچسب‌زنی می‌کنند (یا توسط گروهی از افراد یا به وسیله شبکه‌های عصبی دیگر). سپس داده‌های برچسب زده شده را به عنوان ورودی به شبکه مورد نظر می‌دهند تا شبکه با استفاده از برچسب‌ها یاد بگیرد (وزن‌ها را طوری به دست آورد) که ورودی‌های برچسب نخورده را خودش برچسب بزند. در شکل ۶ نمونه‌ای از داده‌های برچسب زده شده مورد استفاده در این پژوهش را مشاهده کنید.

شکل ۶ نمونه‌ای از مجموعه داده برچسب زده‌شده

۲-۲-۳- معرفی روش‌های برچسب گذاری جهت آشکار سازی چهره در تصویر

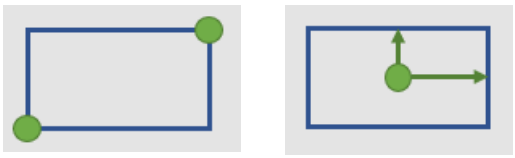
روش‌های مختلفی برای جانمایی چهره در تصویر مورد استفاده قرار می‌گیرد که در ادامه به دو مورد از رایج ترین این روش‌ها اشاره می‌کنیم.

- استفاده از دو گوشه غیرمجاور مستطیل

مطابق شکل ۷ در این روش با استفاده از ابزار های موجود مختصات دو گوشه غیر مجاور از مستطیل در برگیرنده چهره در تصویر توسط خبره علامت گذاری می‌شود.

- استفاده از مرکز و طول و عرض مستطیل

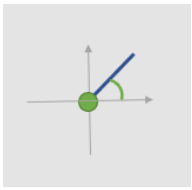
مطابق شکل ۷ در این روش با استفاده از ابزار های موجود مختصات مرکز مستطیل و طول و عرض مستطیل در برگیرنده چهره در تصویر توسط خبره علامت گذاری می‌شود.



شکل 7 جانمایی مستطیلی چهره در تصویر

- استفاده از مرکز و زاویه‌ی نیم‌خط در راستای چهره

مطابق شکل ۸ در این روش با استفاده از ابزارهای موجود مرکز چهره و زاویه نیم خط در راستای چهره توسط خبره علامت گذاری می‌شود.



شکل ۸ جانمایی چهره در تصویر با استفاده از مرکز و زاویه نیم‌خط در راستای چهره

در بحث مقایسه این سه روش، این نکته را می‌توان ذکر کرد که در دو روش اول تنها اطلاعات مکان تقریبی چهره در برچسب نهایی قرار دارد اما در روش سوم علاوه بر مرکز چهره، اطلاعات زاویه قرار گرفتن چهره در تصویر نیز در برچسب گنجانده شده است.

۴-۲-۲- معرفی شبکه‌های عصبی مطرح در حوزه آشکار سازی چهره در تصویر

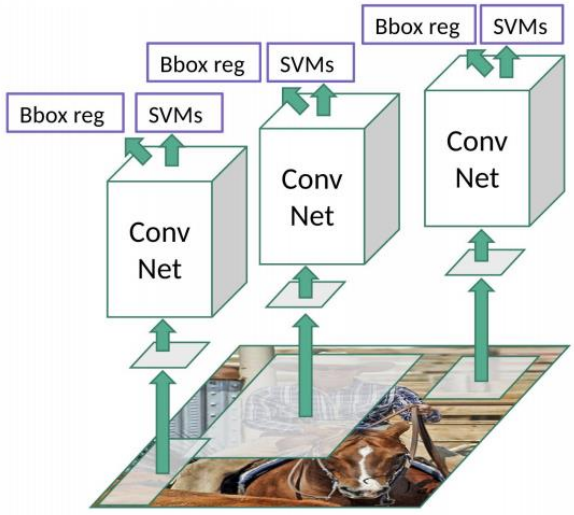
شبکه‌های همگشتی عمیق در ابتدا برای کاربرد دسته بندی تصاویر توسعه یافتند اما در ادامه با توجه به نتایج مناسبی که در آزمایش این شبکه‌ها بدست آمد از این ابزار برای مسئله‌های دیگر پیرامون بینایی کامپیوتر نیز استفاده شده است. در ادامه روند پیشرفت حوزه آشکار سازی چهره که مشابه‌ترین حوزه با مسئله مورد مطالعه در این پژوهش است بررسی می‌شود و چالش‌های رویکردهای متفاوت در این حوزه بررسی می‌شوند.

- الگوریتم R-CNN [۸]

این الگوریتم شامل دو مرحله است به این ترتیب که ابتدا با بهره‌گیری از روش جستوجوی انتخابی[1]،

[1] Selective Search

تعدای ناحیه پیشنهادی از تصویر ورودی استخراج می‌شود و سپس برچسب هر ناحیه توسط یک شبکه همگشتی[1] پیش‌بینی می‌شود. این دو مرحله به صورت مستقل عمل می‌کند و بخش اول تنها ناحیه‌های پیشنهادی را استخراج می‌کند و بخش دوم عمل دسته‌بندی را انجام می‌دهد. با وجود دستیابی به دقت قابل توجه، این الگوریتم بدلیل فراخوانی چند باره شبکه عصبی در این رویکرد، هزینه محاسباتی زیادی دارد و برای بسیاری از کاربردهای زمان حقیقی مناسب نیست. ساختار الگوریتم R-CNN را در شکل ۹ مشاهده می‌کنید.

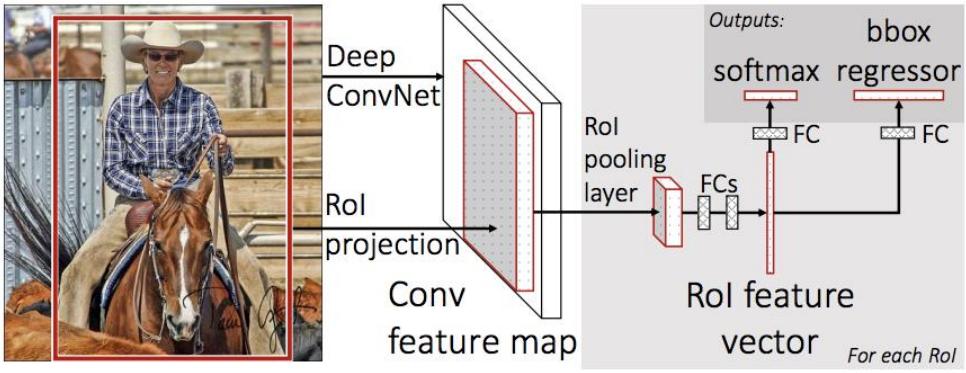


شکل ۹ ساختار شبکه R-CNN

- الگوریتم Fast R-CNN [۹]

به منظور افزایش سرعت الگوریتم R-CNN الگوریتم Fast R-CNN معرفی شد تا از تکرار محاسبات لایه‌های همگشتی جلوگیری شود. در این رویکرد محاسبات لایه‌های همگشتی برای کل تصویر تنها یک بار صورت می‌گرفت، سپس ویژگی‌های مربوط به هر ناحیه پیشنهادی از آن به روش جستوجوی انتخابی[2] استخراج شده و به دسته‌بند وارد می‌شود. ساختار الگوریتم Fast R-CNN را در شکل ۱۰ مشاهده می‌کنید.

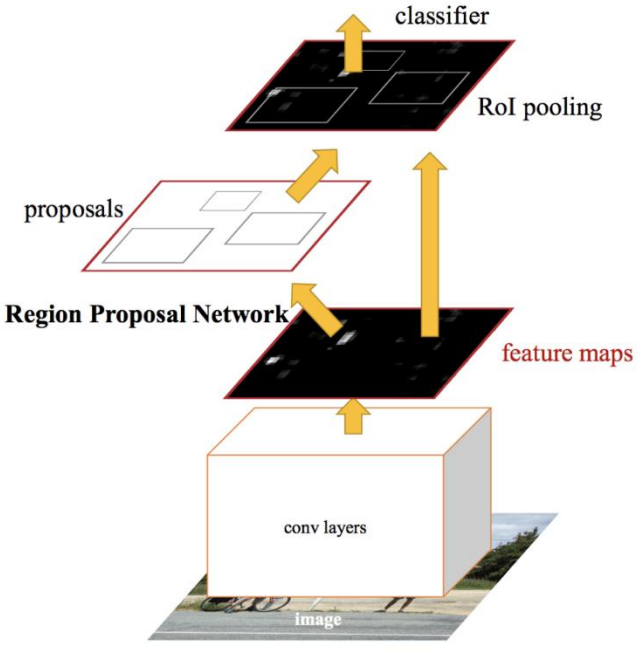
[1] CNN
[2] Selective Search



شکل ۱۰ ساختار شبکه Fast R-CNN

- الگوریتم Faster R-CNN [۱۰]

در نهایت با اضافه کردن بخش آموزش تولید ناحیه‌های پیشنهادی به الگوریتم Fast R-CNN ، این الگوریتم پیشنهاد شد که سرعت و دقت بالاتری را نتیجه داد. ساختار شبکه Faster R-CNN را در شکل ۱۱ مشاهده می‌کنید.



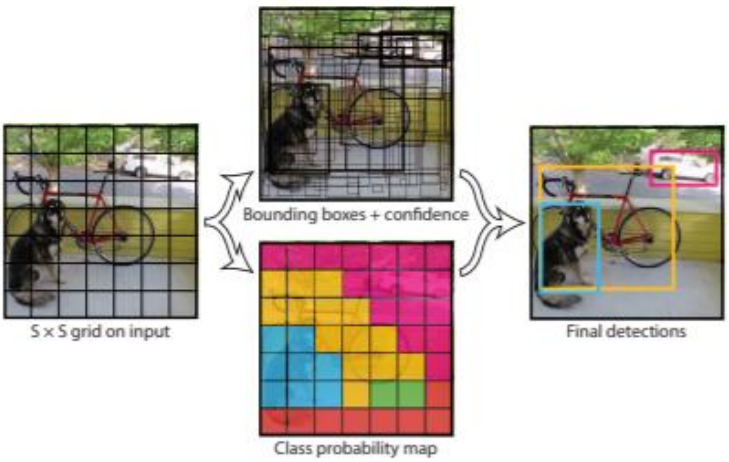
شکل ۱۱ ساختار شبکه Faster R-CNN

الگوریتم‌های مبتنی بر R-CNN دارای دو مرحله تولید ناحیه‌های پیشنهادی و دسته‌بندی هستند. رویکرد دیگری که برای آشکارسازی اشیاء وجود دارد استفاده از پنجره لغزان بجای تولید ناحیه‌های پیشنهادی است. در برداشت اول ممکن است الگوریتم‌های مبتنی بر پنجره لغزان کندتر بنظر برسند اما

می‌توان آن‌ها را به‌گونه‌ای پیاده سازی کرد که با جلوگیری از تکرار محاسبات تکراری مربوط به پیکسل‌های مجاور، به مراتب سریع‌تر از الگریتم‌های مبتنی بر R-CNN عمل کنند.

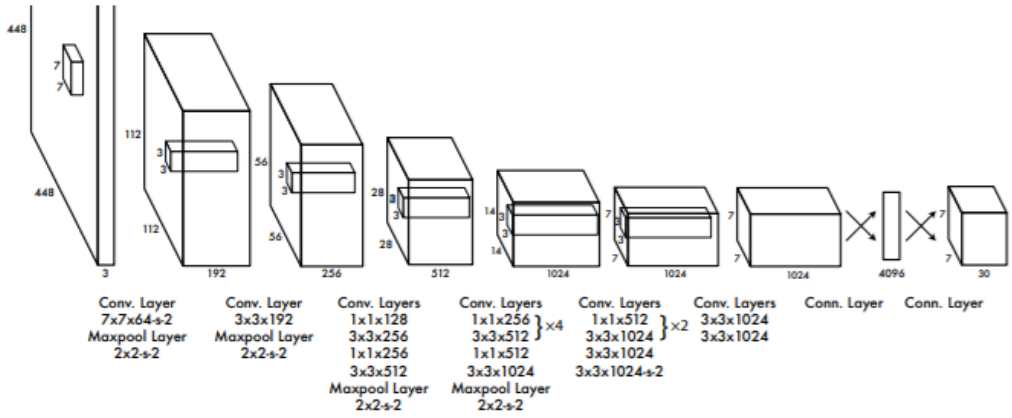
- الگوریتم YOLO [۱۱]

YOLO یکی از نخستین پژوهش‌هایی است که از رویکرد پنجره لغزان استفاده می‌کند. مطابق تصویر۹ در ورودی الگوریتم YOLO تصاویر به تعدادی ناحیه بخش بندی می‌شوند و برای هر ناحیه وجود اشیاء و محدوده آن آموزش داده می‌شود. نحوه عملکرد این الگوریتم را در شکل۱۲ مشاهده می‌کنید.



شکل ۱۲ نمایی از نحوه عملکرد الگوریتم YOLO

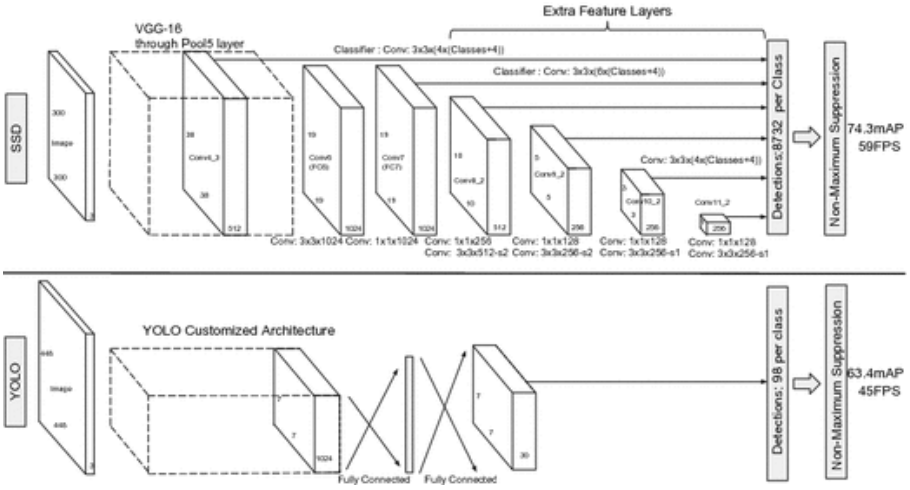
الگوریتم پایه YOLO توانایی پردازش ۴۵ تصـــویر برثانیه را دارد و البته نســـخه کوچک‌تر آن به نام Fast YOLO توانایی پردازش ۱۵۵ تصـویر در ثانیه را دارد سـاختار شـبکه الگوریتم YOLO را در شـکل ۱۳ مشاهده می‌کنید.



شکل ۱۳ ساختار شبکه YOLO

- الگوریتم SSD [۱۲]

الگوریتم SSD توانست با در نظر گرفتن یک ساختار چند مقیاسه علاوه بر افزایش دقت نسبت به YOLO، سرعت قابل قبولی را نیز نتیجه می‌دهد. مقایسه ساختار شبکه الگوریتم SSD با شبکه الگوریتم YOLO را در شکل۱۴ مشاهده می‌کنید.



شکل ۱۴ مقایسه ساختار شبکه الگوریتم SSD و YOLO

- الگوریتم RetinaFace [۱۳]

الگوریتم RetinaFace می‌تواند ۳ کار مختلف بازسازی چهره را همزمان انجام بدهد که عبارتند از: تشخیص چهره، هم‌ترازی دو بعدی چهره و بازسازی سه بعدی چهره. این الگوریتم هر ۳ کار را بصورت همزمان و فقط با یکبار دیدن تصویر و پیاده‌سازی بر یک بستر مبتنی بر یک-دیدار تصویر انجام می‌دهد، به همین دلیل این الگوریتم سرعت بسیار زیادی در آشکارسازی چهره در تصاویر دارد.

۵-۲-۲- چالش‌های رویکرد الگوریتم‌های مبتنی بر پنجره لغزان

در الگوریتم‌های مبتنی بر R-CNN دو مرحله‌ای بودن فرآیند آشکارسازی چهره در تصویر باعث کند شدن فرآیند پیش‌بینی، توسط الگوریتم می‌شود. این مسئله در رویکرد پنجره لغزان به کلی مرتفع می‌شود اما چالش این رویکرد عدم توازن است، به عبارت دیگر تعداد نمونه‌هایی که شامل چهره نیستند به مراتب بیشتر از سایر نمونه‌ها است این مسئله باعث می‌شود که اگر الگوریتم همه نمونه‌ها را هم پس زمینه پیش‌بینی کند در نهایت خطای قابل توجهی متوجه الگوریتم نمی‌شود.

تابع ضرر Focal [۱۴] با افزایش اهمیت نمونه‌های با ضرر زیاد (هرچند با تعداد نمونه‌های کم) میتواند در شرایط عدم توازن عملکرد بهتری را برای شبکه رقم بزند.

در نتیجه‌ی این موارد استفاده از RetinaFace بهترین نتیجه را هم از نظر سرعت و هم از نظر دقت به

دست می‌دهد.

۳-۲- سیستم ردیاب [۱۵]

سیستم‌های ردیاب سیستم‌هایی هستند که یک چهره یا یک شئ را در ویدیو دنبال می‌کنند و با گرفتن مکان اولیه‌ی آن شئ در یک فریم، آن را در تمامی فریم‌های بعدی دنبال می‌کنند و مکان آن در هر فریم را به ما اعلام می‌کنند. این سیستم‌ها استفاده‌های متعددی دارند که تعدادی از آن‌ها عبارتند از: ایجاد تعاملات بین انسان و کامپیوتر، امنیت و نظارت، تحلیل ویدیو با استفاده از تحلیل اشیاء موجود در آن، واقعیت افزوده، کنترل ترافیک، تصویربرداری پزشکی و ویرایش ویدیو. ردیابی اشیاء موجود در ویدیو با توجه به حجم اطلاعاتی که در یک ویدیو وجود دارد ممکن از فرآیند زمان‌بری باشد. همچنین اصولا نیاز به استفاده از یک سیستم تشخیص اشیاء نیز، در کنار آن وجود دارد که این خود به پیچیدگی محاسباتی این گونه سیستم‌ها می‌افزاید.

۱-۳-۲- مقصود در سیستم‌های ردیابی ویدیو

مقصود در ردیابی اشیاء در ویدیو این است که شیء هدف را در فریم‌های متوالی از ویدیو دنبال کرده و یک دنباله از حرکات و مکان شئ در زمان‌های مختلف داشته باشیم. ساخت این دنباله ممکن است در شرایط مختلف کار دشواری باشد. مثلا هنگامی که سرعت حرکت شئ در ویدیو نسبت به نرخ بروز رسانی تصویر[1] زیاد باشد و یا هنگامی که جهت شئ در فریم‌های متوالی عوض شود، مثلا چهره‌ای که با چرخش از تمام رخ به نیم‌رخ تغییر جهت دهد. در اینگونه موارد اصولا سیستم‌های ردیاب یک مدل حرکتی[2] را بکار می‌گیرند که با استفاده از آن بتوانند بهتر حرکات اشیاء در شرایط مختلف را در ویدیو پیش‌بینی کنند.

۲-۳-۲- الگوریتم‌های موجود در سیستم‌های ردیابی ویدیو

برای ردیابی اشیاء در ویدیو از الگوریتم‌هایی استفاده می‌شود که دنباله فریم‌های یک ویدیو را می‌گیرند و با پردازش آن‌ها حرکت اشیاء هدف از هر فریم به فریم بعد را گزارش می‌کنند. در این زمینه الگوریتم‌هایی زیادی وجود دارد که هرکدام مزایا و معایب خود را دارند که آن‌ها را برای کاربردهای مختلف مناسب کرده است. این الگوریتم‌ها را می‌توان به دو دسته تقسیم کرد.

---

[1] Frame per second

[2] Motion model

- الگوریتم‌های بازنمایی و بومی‌سازی هدف

این الگوریتم‌ها اصولا الگوریتم‌هایی از پایین به بالا هستند. این الگوریتم‌ها اصولا ابزارهای زیادی را برای شناسایی شئ در حال حرکت در اختیار ما قرار می‌دهند. شناسایی و ردیابی موفقیت‌آمیز شئ مورد نظر به انتخاب الگوریتم بستگی دارد و باید الگوریتم نهایی بر اساس نیاز انتخاب شود. همچنین پیچیدگی محاسباتی برای این الگوریتم‌ها اصولا پایین بوده.

- ردیابی مبتنی بر هسته[1]

یک روش محلی سازی تکراری برمبنای به حداکثر رساندن یک معیار شباهت.

- ردیابی کانتور[2]

این روش مبتنی برتشخیص مرز شئ عمل می‌کند. به این صورت که به طور مکرر یک کانتور اولیه را که از فریم قبلی به موقعیت جدید آن در فریم فعلی مقداردهی شده است، تغییر می دهند. این رویکرد برای ردیابی کانتور به طور مستقیم با به حداقل رساندن انرژی کانتور با استفاده از گرادیان نزولی، کانتور را تغییر می دهد.

- الگوریتم‌های فیلتر کردن و ارتباط داده‌ها

این الگوریتم‌ها اصولا الگوریتم‌هایی از بالا به پایین هستند که شامل ترکیب اطلاعات قبلی در مورد صحنه یا شئ، پرداختن به دینامیک شی و ارزیابی فرضیه‌های مختلف است. این روش‌ها امکان ردیابی اجسام پیچیده را به همراه تعامل اشیاء پیچیده‌تر مانند ردیابی اجسام در حال حرکت در پشت موانع را فراهم می‌کند.

۴-۲- معرفی سیستم‌های دسته‌بند

در تجزیه و تحلیل دسته‌بندی یا خوشه‌بندی، گروه‌بندی مجموعه‌ای از اشیاء انجام می‌شود. این کار به این صورت است که اشیاء در یک گروه (به نام خوشه) در مقایسه با دیگر دسته‌ها (خوشه‌ها) مشابه‌تر هستند. این وظیفه اصلی داده‌کاوی اکتشافی است و یک روش معمول برای تجزیه و تحلیل داده‌های آماری است که در بسیاری از زمینه‌ها از جمله یادگیری ماشین، تشخیص الگو، تجزیه و تحلیل تصویر، بازیابی اطلاعات،

[1] Kernel-based tracking
[2] Contour tracking

بیوانفورماتیک، فشرده‌سازی داده‌ها و گرافیک کامپیوتری استفاده می‌شود.

تجزیه و تحلیل خوشه‌ای و دسته‌بندی داده خود یک الگوریتم خاص نیست، بلکه روند کلی است و می‌تواند توسط الگوریتم‌های مختلفی به دست آید.

الگوریتم خوشه‌بندی مناسب و تنظیمات پارامترها (از جمله پارامترهایی مانند تابع فاصله مورد استفاده، آستانه تراکم یا تعداد خوشه مورد انتظار) بستگی به تنظیم مجموعه‌داده‌ها توسط فرد و استفاده خاص فرد از نتایج دارد. تجزیه و تحلیل خوشه‌ای یک روش اتوماتیک نیست، بلکه یک فرایند تکراری از کشف دانش یا بهینه‌سازی چند هدفه تعاملی است که شامل آزمایش و شکست است. اغلب لازم است که داده‌های پیش پردازش شده و پارامترهای مدل اصلاح شوند تا نتیجه حاصل، همان نتیجه دلخواه باشد.

۱-۴-۲- تعریف [۱۶]

مفهوم خوشه یا دسته را دقیقا نمی‌توان تعریف کرد، یکی از دلایلش این است که الگوریتم‌های دسته‌بندی زیادی وجود دارد. همه آن‌ها یک قسمت مشترک دارند و آن یک گروه از اشیاء داده است. با این حال، محققان از مدل‌های مختلف خوشه استفاده می‌کنند و برای هر یک از این مدل‌های خوشه، الگوریتم‌های مختلفی را می‌توان ارائه داد. مفهوم یک خوشه، همان‌طور که توسط الگوریتم‌های مختلف یافت می‌شود، به‌طور خاصی در خواص تفاوت دارند. درک این مدل‌های خوشه، کلید فهمیدن تفاوت بین الگوریتم‌های مختلف است. مدل‌های خوشه‌ای معمول عبارتند از:

- مدل‌های متصل: به عنوان مثال، خوشه‌بندی سلسله مراتبی، مدل‌هایی براساس فاصله متصل را ایجاد می‌کند.
- مدل‌های مرکزی: به عنوان مثال، الگوریتم k-means، هر خوشه را با یک بردار متوسط نشان می‌دهد.
- مدل‌های توزیع: خوشه‌ها با استفاده از توزیع‌های آماری، مانند توزیع نرمال چند متغیره که در الگوریتم حداکثر انتظار، استفاده شده‌است.
- مدل‌های تراکم: به عنوان مثال، DBSCAN و OPTICS خوشه را به عنوان مناطق متراکم متصل در فضای داده تعریف می‌کنند.
- مدل‌های زیر فضایی: در biclustering (که به عنوان خوشه مشترک یا خوشه‌ای دو حالت شناخته می‌شود)، خوشه‌ها با هر دو اعضای خوشه و ویژگی‌های مرتبط مدل‌سازی می‌شوند.

- مدل‌های گروهی: برخی از الگوریتم‌ها یک مدل تصحیح شده برای نتایج خود را ارائه نمی‌دهند و فقط اطلاعات گروه‌بندی را ارائه می‌دهند.
- مدل‌های مبتنی بر گراف: یک کلاس، یعنی یک زیر مجموعه از گره‌ها در یک گراف به طوری که هر دو گره در زیر مجموعه با یک لبه متصل می‌شود که می‌تواند به عنوان یک شکل اولیه از خوشه مورد توجه قرار گیرد.
- مدل‌های عصبی: شبکه عصبی غیرقابل نظارت، شناخته شده‌ترین نقشه خود-سازمانی است و معمولاً این مدل‌ها می‌توانند با یک یا چند مدل فوق، مثلا مدل‌های زیر فضایی، زمانی که شبکه‌های عصبی یک مدل تجزیه و تحلیل مؤلفه اصلی و یا یک مدل مستقل تجزیه و تحلیل المان شکل می‌دهند.

خوشه بندی اساساً مجموعه‌ای از خوشه‌ها است که معمولاً شامل تمام اشیاء در مجموعه‌داده‌ها می‌شود. افزون بر این، می‌توان رابطه خوشه‌ها با یکدیگر را مشخص کرد. به عنوان مثال، سلسله مراتب خوشه‌های تعبیه شده در یکدیگر.

خوشه‌بندی را می‌توان براساس شدت تمایز به صورت زیر مشخص کرد:

- خوشه‌بندی سخت: هر شیء متعلق به خوشه است یا نه.
- خوشه‌بندی نرم (همچنین خوشه بندی فازی[1]): هر شیء به مقدار مشخصی به هر یک از خوشه‌ها متعلق است (به عنوان مثال، احتمال عضویت در هر خوشه)

همچنین امکان تمایز دقیق‌تر وجود دارد، مثلاً:

- خوشه‌بندی جداسازی دقیق (پارتیشن‌بندی)[2]: هر شیء دقیقاً به یک خوشه متعلق است.
- خوشه‌بندی جداسازی دقیق با ناپیوستگی[3]: اشیاء می‌توانند به هیچ خوشه‌ای تعلق نداشته باشند.

خوشه‌بندی هم‌پوشانی[4] (همچنین: خوشه‌بندی جایگزین، خوشه‌بندی چندگانه): اشیاء ممکن است متعلق به بیش از یک خوشه باشد؛ معمولاً خوشه‌های سخت را شامل می‌شود.

خوشه‌بندی سلسله مراتبی[5]: اشیایی که متعلق به خوشه فرزند هستند، متعلق به خوشه پدر و مادر هم

---

[1] Fuzzy clustering
[2] Strict Partitioning
[3] Strict partitioning with outliers
[4] Overlapping clustering
[5] Hierarchical clustering

هستند.

خوشه‌بندی زیر فضا: یک نوع خوشه‌بندی همپوشانی است، که در یک زیر فضای منحصر به فرد تعریف شده و انتظار نمی‌رود که خوشه‌ها باهم هم‌پوشانی داشته باشند.

۵-۲- مراحل عمومی طراحی سامانه‌های آشکار سازی چهره در تصویر

این سامانه‌ها اغلب از ۵ مرحله برای آشکار سازی چهره در تصویر استفاده می‌نمایند.

- **جمع آوری داده‌ها:** اولین اقدام برای حل مسئله با رویکردهای مبتنی بر یادگیری ماشین جمع آوری داده‌های متناسب با صورت مسئله است، که در مورد حوزه بینایی ماشین این داده ها به صورت تصاویر جمع آوری می‌شوند.
- **برچسب زدن داده‌ها:** این مرحله اولین قدم پس از جمع آوری داده‌ها به شمار می‌رود و در آن اطلاعات مورد هدف جهت پیش‌بینی، توسط یک خبره از تصاویر استخراج و به عنوان برچسب به داده‌ها اضافه می‌شود. با توجه به اینکه در این پژوهش اطلاعات هدف شناسایی و دسته‌بندی چهره اشخاص است، برای برچسب زدن داده‌ها باید از میان روش‌ها پیش‌تر ذکر شده روشی را انتخاب کرد که محدوده‌ی چهره را به خوبی مشخص کند و بتوانیم مرکز چهره را نیز برای یافتن مسیر حرکت چهره‌ها در تصویر و یافتن دنبله‌های هر چهره در ویدیو بدست بیاوریم، به همین جهت در این پژوهش از روش استفاده از دو گوشه غیرمجاور مستطیل جهت برچسب زدن تصاویر استفاده شده است. در شکل۱۵ نمونه‌ای از تصاویر مجموعه داده را مشاهده می‌کنید.

شکل ۱۵ نمونه‌ای از تصاویر مجموعه داده

- **پیش‌پردازش**: آماده‌سازی عکس‌های مجموعه‌داده ورودی برای ورودی دادن به شبکه که خود شامل ۳ بخش است:
  - **یکسان سازی اندازه تصاویر**: با توجه به اینکه ورودی شبکه یک اندازه واحد دارد و تصاویر چهره‌های استخراج شده از فریم‌های ویدیو با توجه به زاویه‌ی صورت نسبت به دوربین و فاصله شخص از دوربین دارای اندازه‌های متفاوت هستند در اولین قدم اندازه‌ی همه‌ی تصاویر چهره به اندازه‌ی هدف تبدیل می‌شود.
  - **برخی پیش پردازش‌های لازم**: در مرحله‌ی بعد از ماژول preprocess_input() از شبکه‌ی resnet استفاده می‌شود که معمولا همه‌ی پیش پردازش‌های لازم را (از جمله تنظیم رنگ و روشنایی و ...) را برای ما انجام می‌دهد و تصویر را آماده‌ی ورودی دادن به شبکه می‌کند.
  - **افزودن بعد چهارم:** در این مرحله یک بعد چهارم به تصاویر اضافه می‌شود تا برای پردازش دسته‌ای آماده باشند.
- **الگوریتم تشخیص چهره**: استفاده از الگوریتم RetinaFace برای تشخیص چهره‌های افراد موجود در هر تصویر.
- **سیستم ردیاب**: پیدا کردن دنباله‌های تصاویر چهره‌ی افراد به واسطه‌ی خواندن ویدیو بصورت فریم فریم و اجرای الگوریتم تشخیص چهره بر روی هر فریم و استفاده از تصاویر چهره‌ی خروجی از آن. با توجه به نوع مسئله برای ردیابی چهره‌های افراد ما از میان الگوریتم‌های ذکر شده از الگوریتمی از دسته‌ی الگوریتم‌های بازنمایی و بومی‌سازی هدف استفاده می‌کنیم.
- **شبکه عصبی با وزن‌های از قبل آموخته**: پیاده‌سازی و استفاده از شبکه‌ی عصبی با وزن‌های از قبل آموخته[1].
- **تشخیص و دسته‌بندی**: با توجه به نوع داده‌های ورودی به سیستم دسته‌بند، دسته‌بندی دنباله‌های تصاویر با استفاده از یک دسته‌بند از نوع سلسله مراتبی صورت می‌گیرد، به صورتی که همه‌ی دنباله‌های تصاویر چهره‌ی متعلق به یک فرد خاص در یک دسته واحد قرار بگیرند. در ادامه بیشتر در این مورد توضیح می‌دهیم.
- **تنظیم دقیق‌تر[2] شبکه‌ی عصبی:** با استفاده از مجموعه داده‌ای که داریم و برچسب‌هایی که برای آن گذاشته‌ایم شبکه‌ی عصبی را تنظیم دقیق می‌کنیم و مجدد فرآیند تشخیص و دسته‌بندی را انجام

[1] Pre-trained network
[2] Fine-Tune

می‌دهیم تا ببینیم که آیا دقت شبکه عصبی در بدست آوردن اطلاعات از تصاویر چهره‌ها بهبود یافته یا خیر.

۶-۲- نتیجه‌گیری

تاکنون تحقیقات زیادی در حوزه آشکار سازی و دسته‌بندی چهره‌ی اشخاص در تصاویر انجام شده است و به نتایج قابل قبولی نیز دست پیدا شده است. ما در این تحقیق قصد بررسی این موضوع را داریم که آیا با استفاده از تنظیم دقیق‌تر شبکه عصبی می‌توانیم باعث افزایش دقت در دسته‌بندی تصاویر چهره‌های افراد موجود در ویدیو شویم یا خیر.

فصل ۳:
روش تحقیق

۱-۳- مقدمه

در این پروژه قصد داریم که با استفاده ابزارهای بینایی کامپیوتر و شبکه‌های عصبی بتوانیم با سرعت و دقت مناسب، شخصیت‌های حاضر در ویدیو را تشخیص دهیم و حرکات و زمان حضور آن‌ها در ویدیو را تحلیل کنیم و در انتها نیز شخصیت‌ها را دسته‌بندی کنیم. در انتهای کار تلاش ما این است که بصورت خود نظاره‌گر[1] این سیستم را بتوانیم بهبود نیز بدهیم.

۲-۳- ساختار سیستم استفاده شده

با توجه به توضیحاتی که در فصل قبل درباره معرفی و مقایسه‌ی شبکه‌های عصبی و سیستم‌های ردیابی مطرح داده شد، تصمیم گرفتیم سیستمی که پیاده سازی می‌کنیم دارای ویژگی‌های زیر باشد:

- استفاده از الگوریتم آشکارساز چهره RetinaFace برای استخراج محدوده چهره از فریم‌های ویدیو
- پیاده سازی یک سیستم ردیابی ویدیو بر مبنای الگوریتم‌های ردیابی کانتور و محدوده چهره‌های استخراج شده توسط الگوریتم آشکارساز چهره و استخراج دنباله‌های چهره [۱۷]
- استفاده از تکنیک یادگیری انتقالی (استفاده از وزن‌های شبکه‌های از قبل آموزش دیده) برای پیاده‌سازی شبکه عصبی مورد استفاده در استخراج ویژگی از تصاویر چهره
- استفاده از یک دسته‌بند بر مبنای دسته‌بندی سلسله مراتبی تجمعی جهت دسته بندی دنباله تصاویر چهره‌های استخراج شده [۱۸]
- استفاده از شبکه عصبی Siamese برای تنظیم دقیق‌تر شبکه عصبی مورد استفاده در استخراج ویژگی از تصاویر چهره [۱۹]
- استفاده از تابع ضرر سه‌گانه جهت تنظیم دقیق‌تر شبکه و استفاده از سیگنال‌های مثبت و منفی در فرآیند یادگیری [۲۰]

۱-۲-۳- الگوریتم‌های استخراج چهره از فریم‌های ویدیو

برای این کار هر دو ابزار YOLOFace و RetinaFace از دقت بالا و مناسبی برخوردار هستند و به خوبی تصاویر چهره را استخراج می‌کنند. به همین دلیل یک بخش ۱۲ ثانیه‌ای از مجموعه‌داده‌ی مورد

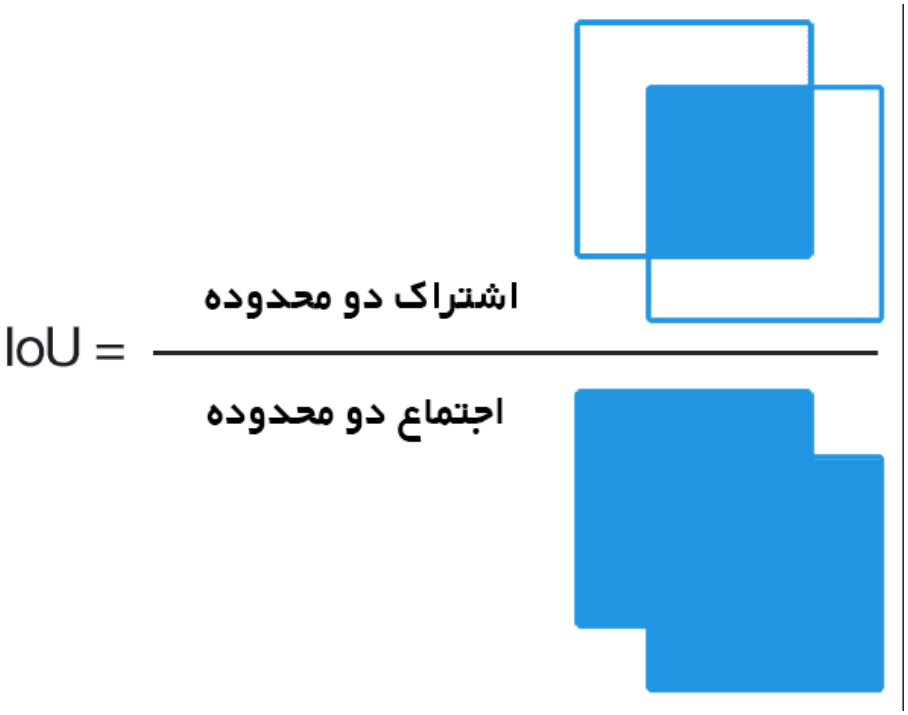
[1] Self-Supervised

آزمایش را با برچسب‌های محدوده‌ی چهره برچسب گذاری می‌کنیم و با معیاری به نام iou[1] یا اشتراک بر اجتماع [۲۱] آن‌ها را مقایسه می‌کنیم تا ببینیم کدام یک از این دو ابزار برای سیستم ما مناسب‌تر هستند. این معیار به شرح رابطه‌ی زیر تعریف می‌شود:

IoU = (مساحت اشتراک دو محدوده) / (مساحت اجتماع دو محدوده)

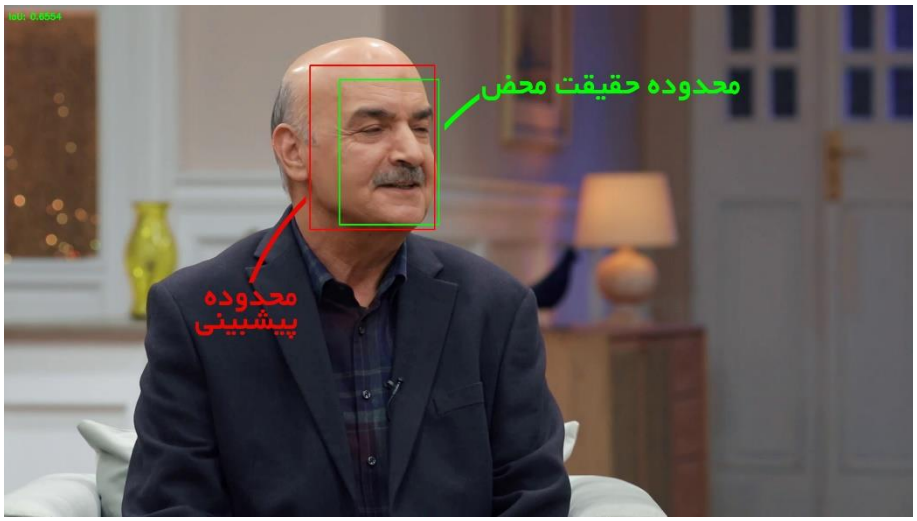
که در شکل ۱۶ با جزئیات بیشتر قابل مشاهده است. در تعریف این معیار دو اصطلاح استفاده می‌شود:

- محدوده حقیقت محض[2]: که در واقع همان محدوده چهره‌ی مشخص شده در مجموعه‌داده است.
- محدوده پیشبینی[3]: که در واقع محدوده چهره‌ی پیشبینی شده توسط ابزار استخراج چهره است.



شکل ۱۶ شرح رابطه iou

[1] Intersection over union
[2] Gound truth bounding box
[3] Predicted bounding box



شکل ۱۷ تصویری از مجموعه داده و محدوده‌های برچسب گذاری شده و پیشبینی شده

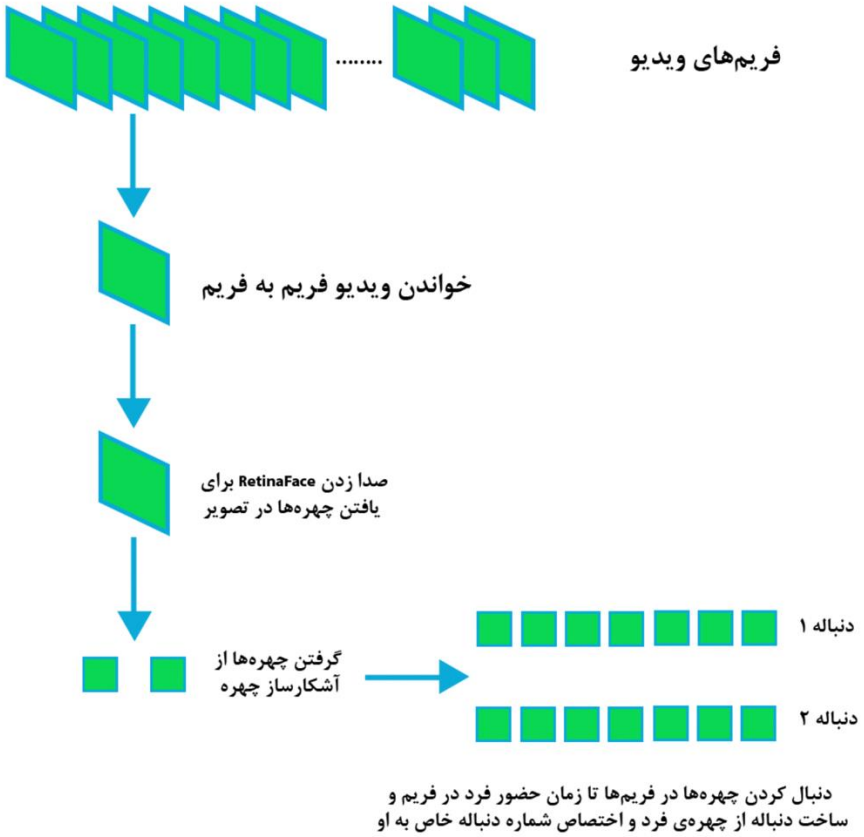
پس از آزمایش هر دو ابزار و بدست آوردن مقدار iou میانگین برای هر کدام روی این محموعه‌داده نتایج بدست آمده برای آن‌ها نشان می‌دهد که ابزار RetinaFace برای پژوهش ما مناسب‌تر است. این نتایج را می‌توانید در جدول ۱ مشاهده کنید.

جدول ۱ نتایج IoU ابزارها بر روی مجموعه‌داده

ابزار	مقدارIoU میانگین
RetinaFace	۰.۶۶۴۵۳
YOLOFace	۰.۵۶۸۵۲

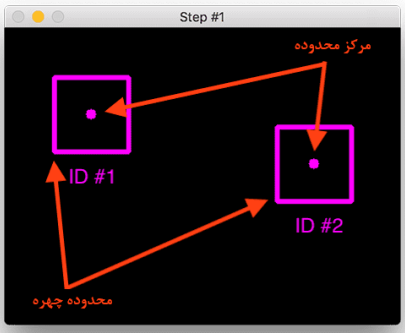
۲-۲-۳- ساختار عمومی سیستم ردیابی ویدیو

برای این امر از سیستم ردیابی ویدیو بر مبنای الگوریتم‌های ردیابی کانتور استفاده شده است. سیستم ردیابی ویدیو در این پژوهش بخش اولیه و ورودی سیستم است و فایل ویدیو را به عنوان ورودی دریافت می‌کند و تصاویر محدوده چهره‌ها در هر فریم را استخراج می‌کند و در قالب دنباله‌هایی از تصاویر چهره خروجی می‌دهد. نمای کلی سیستمی که برای این امر استفاده شده است را می‌توانید در شکل ۱۸ مشاهده کنید.



شکل ۱۸ ساختار کلی سیستم ردیابی ویدیو

همان‌طور که در شکل ۱۹ نیز مشاهده می‌کنید این سیستم به این شکل عمل می‌کند که ویدیو را بصورت فریم به فریم می‌خواند و برای هر فریم یک باز آشکارساز چهره را صدا می‌زند تا درصورت وجود چهره در تصویر محدوده‌ی آن را مشخص کند و به ردیاب بدهد. سپس ردیاب به ازای هر چهره با استفاده از محدوده چهره دریافتی از آشکارساز، مرکز آن محدوده را محاسبه می‌کند و بر اساس مرکز آن تصمیم می‌گیرد که آیا این چهره به دنباله‌ای تعلق دارد یا خیر؟ اگر به دنباله‌ای تعلق داشته باشد که آن دنباله با این تصویر بروز می‌شود و اگر به هیچ دنباله‌ای تعلق نداشته باشد برای آن دنباله‌ای جدید ساخته می‌شود و یک شناسه‌ی یکتا به آن اختصاص داده می‌شود.



شکل ۱۹ محدوده‌ها و مراکز چهره‌ها و اختصاص شناسه یکتا

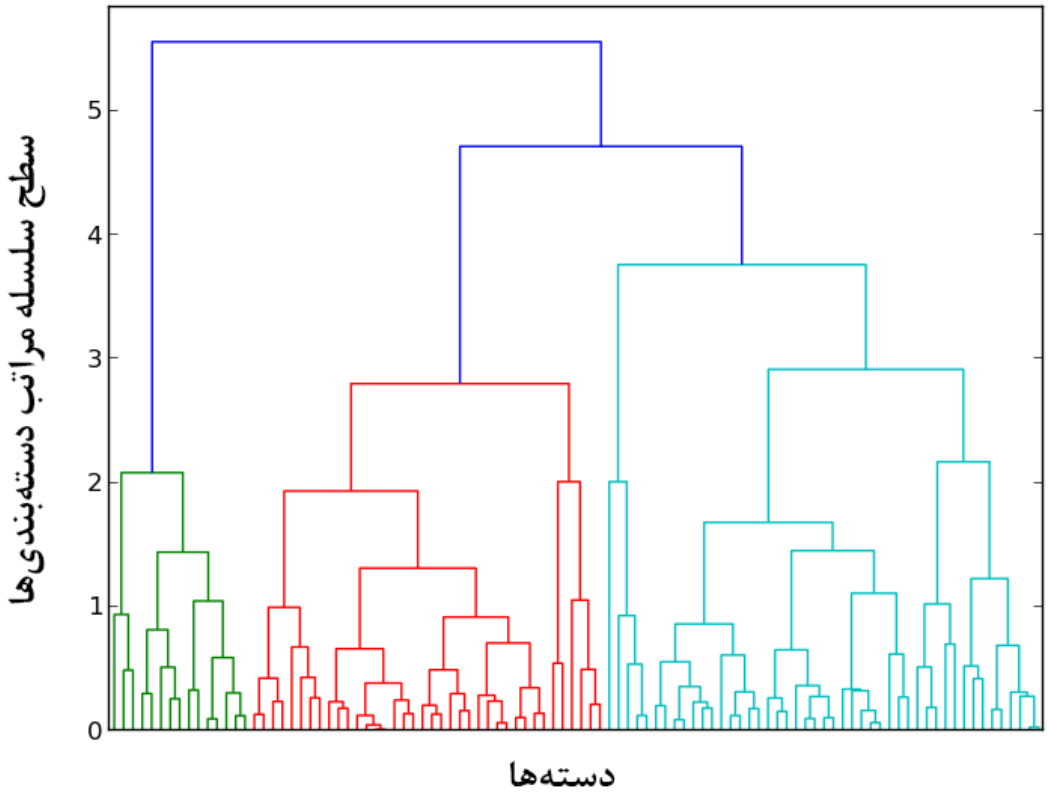
با توجه به تغییراتی که در زاویه‌ی دوربین و محیط ممکن است رخ دهد گاهی ممکن است خطاهایی در سیستم ردیاب رخ دهد و مشکلاتی از قبیل: ادغام دو دنباله متفاوت باهم و مشاهده نویز و داده‌های غیر مرتبط در یک دنباله داشته باشیم. همان‌طور که در ادامه بیشتر توضیح خواهیم‌داد، دقت در استخراج دنباله‌ها برای ما مهم است و سیگنال‌های مثبت و منفی‌ای که برای تنظیم دقیق‌تر شبکه نیاز داریم را از همین دنباله‌ها استخراج خواهیم کرد.

بنابراین قبل از شروع به کار سیستم ردیاب از یک سیستم تشخیص صحنه استفاده خواهیم کرد که ویدیو را بر اساس تغییر صحنه و زاویه‌ی دوربین به چند بخش مختلف تقسیم می‌کند که در هرکدام از این بخش‌ها صحنه و زاویه‌ی دوبین کاملا ثابت است و سپس روی هرکدام از این بخش‌ها سیستم ردیاب را اجرا می‌کنیم. باید توجه داشت که این سیستم تشخیص صحنه در ویدیوهایی مانند ویدیو دوربین مدار بسته که صحنه و زاویه دوربین ثابت است تاثیری ندارد و فقط در مواردی که صحنه و زاویه دوربین زیاد عوض می‌شود دقت ما را بهبود می‌دهد.

۳-۲-۳- سیستم دسته‌بندی [۲۲]

برای دسته‌بندی دنباله تصاویر چهره‌های استخراج شده در این سیستم نیاز به یک سیستم دسته‌بندی داریم. که چون دنباله‌ها برخی بسیار نزدیک به هم هستند (دنباله‌های مربوط به یک فرد یکسان) و برخی به نسبت قبلی‌ها بسیار دور از هم هستند (دنباله‌های مربوط به دو فرد مختلف) از یک دسته‌بند سلسه مراتبی استفاده می‌کنیم و از بردار ویژگی‌های چهره‌های هر دنباله، بردار ویژگی میانگین می‌گیریم و سپس این بردار ویژگی‌های میانگین هر دنباله را به سیستم دسته‌بندی می‌دهیم. در شکل ۲۰ ساختار کلی سیستم دسته‌بند

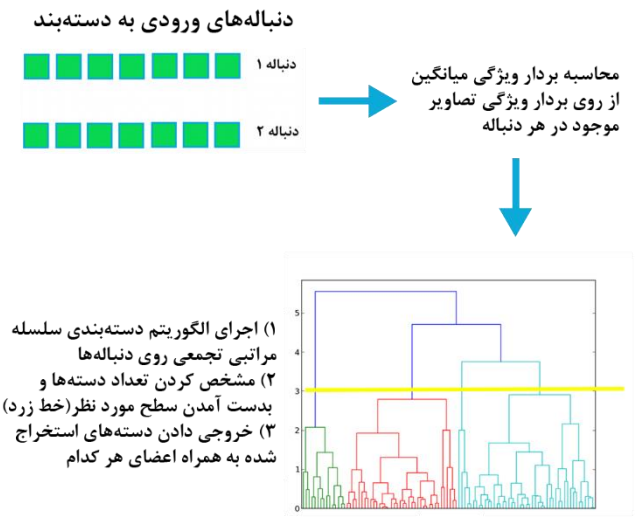
سلسله مراتبی [۲۳] و در شکل ۲۱ ساختار ورودی‌های سیستم ما به دسته‌بند قابل مشاهده است.



شکل ۲۰ ساختار کلی دسته‌بند سلسله مراتبی

همان‌طور که پیشتر ذکر شد در این پژوهش از یک سیستم دسته‌بند سلسله مراتبی تجمعی استفاده می‌کنیم. رویکرد الگوریتم‌های دسته‌بندی سلسله مراتبی تجمعی، یک رویکرد از پایین به بالا می‌باشد. به این صورت که با شروع از پایین، در هر مرحله دو دسته با یکدیگر تجمیع شده و یک دسته جدید تشکیل می‌دهند. دسته‌های جدید در سطح‌های بالاتر قرار گرفته و این روند تکرار می‌شود.

دسته‌بندی تجمعی با یک دسته به ازای هر مشاهده شروع شده و در هر مرحله، دو دسته که کمترین تفاوت را با یک‌دیگر دارند تجمیع شده و یک دسته بزرگتر را تشکیل می‌دهند. این کار باعث می‌شود تا در سطح بالاتر یک دسته کمتر وجود داشته باشد و تا زمانی که تعداد دسته‌ها به یک برسد این کار ادامه پیدا می‌کند.



شکل ۲۱ ساختار ورودی‌های سیستم ما به دسته‌بند سلسله مراتبی و فرآیند طی شده در سیستم برای دسته‌بندی دنباله‌ها

همان‌طور که در شکل ۲۱ قابل مشاهده است، در این پژوهش قبل از دسته‌بندی دنباله‌های تصاویر چهره، در هر دنباله از بردارهای ویژگی تصاویر آن دنباله میانگین می‌گیریم و این بردار ویژگی میانگین را، بردار ویژگی نماینده‌ی آن دنباله قرار می‌دهیم و بر اساس همین بردار ویژگی میانگین دنباله‌ها را دسته‌بندی می‌کنیم.

۴-۲-۳- ساختار عمومی شبکه عصبی Siamese

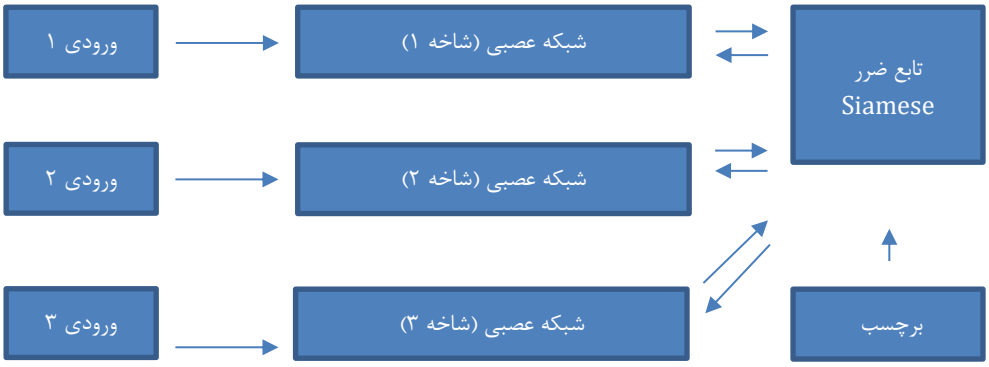
شبکه Siamese نوعی معماری شبکه است که شامل دو یا چند زیرشبکه یکسان است که برای تولید بردارهای ویژگی برای هر ورودی و مقایسه آن‌ها استفاده می‌شود. شبکه‌های Siamese را می‌توان برای موارد استفاده مختلف مانند شناسایی موارد تکراری[۱]، یافتن ناهنجاری‌ها[۲] و تشخیص چهره[۳] اعمال کرد. یادگیری در شبکه‌های Siamese را می‌توان با از تابع ضرر سه‌گانه یا تابع ضرر متضاد انجام داد. در این پژوهش ما قصد داریم از یک تابع ضرر سه‌گانه استفاده کنیم. برای یادگیری با تابع ضرر سه‌گانه، یک بردار پایه (تصویر لنگر)[۴] با یک بردار مثبت (تصویر واقعی)[۵] و یک بردار منفی (تصویر نادرست)[۶] مقایسه می‌شود. بردار منفی یادگیری را در شبکه اعمال می‌کند، در حالی که بردار مثبت مانند یک تنظیم کننده عمل می

[1] Duplicate detection
[2] Finding anomalies
[3] Face recognition
[4] Anchor
[5] Positive
[6] Negative

کند. در این پژوهش ما قصد داریم از یک شبکه‌ی Siamese برای تشخیص چهره استفاده کنیم.

بطور کلی ساختار عمومی شبکه‌ی Siamese شامل دو بخش اصلی است:

- بخش شبکه عصبی، که وظیفه آن استخراج بردارهای ویژگی از تصاویر است و به تعداد ورودی‌های همزمان زیرشبکه‌ها به شبکه‌ی Siamese تکرار می‌شود و برای همه‌ی زیرشبکه‌ها دارای ساختار و وزن‌های مشترک است.
- بخش محاسبه تابع ضرر شبکه Siamese که در این پژوهش تابع ضرر از جنس تابع ضرر سه‌گانه است.



شکل ۲۲ ساختار عمومی شبکه‌های عصبی Siamese

در شکل ۲۲ نمای کلی از ساختار شبکه‌های عصبی Siamese را مشاهده می‌کنید. در ادامه به جزئیات پیاده سازی هر یک از این بخش‌ها، در روش حل مسئله خودمان می‌پردازیم.

۵-۲-۳- معرفی ساختار بخش ورودی‌های شبکه

همانطور که پیش‌تر ذکر شد ما از شبکه عصبی Siamese استفاده می‌کنیم تا بتوانیم چند ورودی را مقایسه کنیم و دقت شبکه‌ی عصبی‌ای که برای استخراج بردارهای ویژگی از تصاویر از آن استفاده می‌کنیم را بهبود دهیم. در این پژوهش هدف ما از تنظیم دقیق‌تر شبکه‌ی عصبی این است که بردارهای ویژگی‌ای که از تصاویر برای ما استخراج می‌کند به شکلی بهبود یابند که الگوریتم دسته‌بند بتواند با دقت بالاتری

دنباله‌های تصاویر را دسته‌بندی کند.

به همین جهت هر سه ورودی ما به شبکه باید به شکلی باشند که تصویر هر فردی که به عنوان تصویر پایه به شبکه داده می‌شود برای آن یک تصویر از همان فرد به عنوان جفت مثبت و یک تصویر از فردی دیگر به عنوان جفت منفی نیز به شبکه داده شود تا تابع ضرر Siamese بتواند با در نظر گرفتن این دو سیگنال مثبت و منفی به درستی عمل کند و وزن‌های شبکه را به سمت مطلوب ما ببرد.

شکل ۲۳ نمای کلی چیدمان و انتخاب ورودی‌های شبکه عصبی

در شکل ۲۳ می‌توانید نمای کلی از چیدمان ورودی‌های شبکه را ببینید. در ادامه به جزئیات انتخاب تصاویر ورودی‌ها و انتخاب جفت‌ها در روش حل مسئله خودمان می‌پردازیم.

برای انتخاب تصویری که به عنوان جفت مثبت به شبکه داده می‌شود یک تصویر از همان دنباله‌ی تصاویر تصویر پایه را بصورت تصادفی انتخاب می‌کنیم.

برای انتخاب تصویری که به عنوان جفت منفی به شبکه داده می‌شود یک تصویر را با طی کردن حلقه‌ی زیر انتخاب می‌کنیم:

- ابتدا بررسی می‌کنیم تا ببینیم آیا در همان فریم از ویدیو چهره‌ی دیگری نیز حضور داشته یا خیر؟ در صورت حضور همزمان دو چهره در یک فریم این یک سیگنال قطعی منفی است و این دو تصویر چهره متعلق به یک فرد نیستند. در صورت عدم وجود تصویر با این مشخصات مورد بعد را بررسی می‌کنیم.

شکل ۲۴ حضور چهره‌ی دیگر در فریم

- بررسی می‌کنیم تا ببینیم آیا دنباله تصاویری وجود دارد که در حداقل چند فریم با دنباله تصاویر تصویر پایه حضور هم زمان[1] داشته باشد یا خیر؟ در صورت وجود داشتن چنین دنباله‌ای این یک سیگنال منفی قطعی بوده و این دو دنباله به دلیل حضور هم‌زمان در حداقل یک فریم از ویدیو متعلق به یک فرد نیستند. پس یک تصویر از آن را بصورت تصادفی انتخاب کرده و به عنوان جفت منفی به شبکه می‌دهیم. در صورت عدم وجود تصویر با این مشخصات مورد بعد را بررسی می‌کنیم.
- درصورت عدم رخداد هیچ‌کدام از موارد بالا یک دنباله از دسته‌های دیگر را بصورت تصادفی

[1] Co-occurrence

انتخاب کرده و یک تصویر بصورت تصادفی از میان تصاویر آن‌ها به عنوان جفت منفی انتخاب می‌کنیم. این تصویر با توجه به بالا بودن دقت بخش‌های قبلی سیستم در استخراج دسته‌ها و تصاویر با دقت قابل قبولی یک سیگنال منفی مناسب می‌باشد.

۶-۲-۳- معرفی ساختار بخش شبکه عصبی

همانطور که پیش‌تر ذکر شد، وظیفه اصلی این بخش از شبکه استخراج بردارهای ویژگی از تصویر است. همچنین برای استفاده از تکنیک یادگیری انتقالی ناچار به استفاده از شبکه‌های از پیش‌طراحی شده و یا بخشی از آن‌ها هستیم. مسئله ما نیازمندی‌هایی دارد که این نیازمندی‌ها به ما کمک می‌کند که بتوانیم از میان شبکه‌های معروف، مورد مناسبی را انتخاب کنیم:

این نیازمندی‌ها عبارت‌اند از:

- سرعت بالا، جهت کاربردهای زمان حقیقی
- داشتن خروجی مطلوب در عین محاسبات بهینه
- داشتن وزن‌های بخوبی بهینه شده روی مسئله‌های استخراج اطلاعات از تصویر چهره یا مسئله‌های عمومی، جهت استفاده از تکنیک یادگیری انتقالی
- دقیق بودن وزن‌های اولیه به دلیل اینکه شبکه باید دقت اولیه بالایی داشته باشد تا بتواند سیگنال‌های منفی و مثبت مورد نیاز در تنظیم دقیق شبکه را با دقت مطلوبی و بصورت خود-نظاره‌گر[1] و فقط با در دست داشتن اطلاعات مربوط به دنباله‌های چهره‌ها[2] و بردارهای ویژگی اولیه‌ی آن‌ها به‌دست بیاورد.

با توجه به نیازمندی‌های گفته شده، به سراغ شبکه‌های آموزش دیده بر روی مسئله ۱۰۰۰ دسته‌ای ImageNet [۲۴] رفتیم. مسئله ImageNet بعلت فراوانی در داده، و همچنین دسته‌های متنوع مسئله عمومی خوبی جهت استفاده در کاربردهای تحقیقاتی دیگر بشمار می‌رود.

نکته قابل ذکر این است که شبکه‌های آموزش دیده روی مسئله ImageNet، بدلیل پیچیدگی بالای مسئله و حجم بسیار زیاد مجموعه‌داده اولیه که برای آموزش وزن‌های شبکه به کار رفته است، عموما شبکه‌های عمیق و بسیار عمیقی هستند که بسیار دقیق وزن‌های آن‌ها تنظیم شده است و از تعداد لایه‌های

[1] Self-supervised
[2] Track Supervised

زیاد در پیاده‌سازی شبکه استفاده کرده‌اند که باعث می‌شود تنظیم دقیق آن‌ها با حجم داده‌ی بسیار کم (نسبت به حجم داده در مسئله‌ی ImageNet) کاری بسیار دشوار و تقریبا غیر ممکن باشد و دست زدن به وزن‌های شبکه‌ی اصلی فقط باعث برهم ریختن آن شود.

بنابراین باید به ناچار در صورت انتخاب هر کدام از شبکه‌های از قبل آموزش دیده روی مسئله ImageNet باید لایه‌های شبکه‌ی اصلی را با وزن‌های از قبل آموزش دیده ثابت نگه‌داریم و فقط بخش فوقانی شبکه که حاوی برچسب‌های مسئله‌ی ImageNet است را از آن جدا کنیم و چند لایه جهت یادگیری و تنظیم دقیق‌تر شبکه به آن اضافه کنیم که با وزن‌های این لایه‌ها را بر اساس مجموعه‌داده‌ای که ساخته‌ایم آموزش خواهیم داد.

انتخاب شبکه‌ی ResNet۵۰ [۲۵] می‌تواند انتخاب خوبی باشد. این شبکه هم در محاسبات و سرعت عمل بهینه است و هم عملکرد مناسبی در مسئله‌ی ImageNet به ثبت رسانده است.

در نهایت با استفاده از شبکه‌ی ResNet۵۰ به عنوان شبکه‌ی پایه و اضافه کردن یک لایه Flatten و چند لایه Dense و BatchNormalization به آن، دقت خوبی در دسته‌بندی دنباله‌های تصاویر با استفاده از بردارهای ویژگی استخراج شده توسط شبکه از روی تصاویر به دست می‌آید که آزمایش‌ها نشان می‌دهد که شبکه مناسبی برای استخراج بردارهای ویژگی از تصاویر است. ساختار نهایی شبکه طراحی شده را در شکل ۲۵ مشاهده می‌کنید.

شکل ۲۵ ساختار شبکه طراحی شده

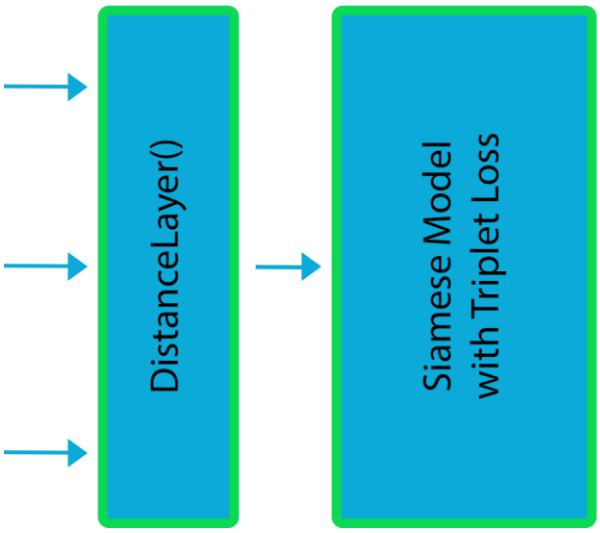
خروجی این شبکه به ازای یک تصویر با ابعاد $W \times H \times 3$ یک بردار ویژگی به طول ۲۵۶ خواهد بود که ویژگی‌های محاسبه شده به ازای هر تصویر ورودی است و برای دسته‌بندی به دسته‌بند داده می‌شود. همچنین از خروجی آن برای تنظیم دقیق‌تر شبکه نیز استفاده خواهیم کرد که در آن‌صورت به تابع ضرر Siamese داده می‌شود.

۷-۲-۳- معرفی ساختار بخش تابع ضرر Siamese

همانطور که پیش‌تر ذکر شد، وظیفه این بخش محاسبه تابع ضرر از سه ورودی دریافتی برای تنظیم دقیق‌تر شبکه و کمک به یادگیری وزن‌های جدید است. اگر بخواهیم دقیق‌تر به این مسئله بپردازیم در واقع این بخش یک ورودی سه‌گانه از بردارهای ویژگی (بردار پایه، بردار جفت مثبت، بردار جفت منفی) دریافت می‌کند و فاصله‌ی بین بردار پایه و بردار جفت مثبت و همچنین فاصله‌ی بین بردار پایه و بردار جفت منفی را محاسبه کرده و با استفاده از تابع ضرری از جنس تابع ضرر سه‌گانه در یادگیری وزن‌های جدید شبکه عصبی و تنظیم دقیق‌تر آن به ما کمک می‌کند.

این بخش شامل دو زیربخش اصلی است:

- لایه محاسبه فاصله، که یک لایه پیاده‌سازی شده است که کار آن این است که با دریافت هر سه بردار ویژگی، فاصله‌ی بردار پایه از بردار جفت مثبت و جفت منفی را محاسبه کند و خروجی بدهد.
- لایه مدل Siamese، که لایه‌ی اصلی در یک شبکه Siamese است، در واقع یک لایه نیست و یک مدل است که با گرفتن فاصله بردارها از لایه‌ی محاسبه فاصله، تابع ضرر (در پژوهش ما تابع ضرر سه‌گانه) را محاسبه می‌کند. این لایه درواقع مدل اصلی شبکه Siamese است و حلقه‌ی یادگیری این شبکه در همین لایه پیاده‌سازی شده است و درواقع بر اساس تعریفTensorFlow به هنگام آموزش شبکه، دربرگیرنده‌ی کل لایه‌های شبکه عصبی است.



شکل ۲۶ ساختار بخش تابع ضرر Siamese

۸-۲-۳- معرفی تابع ضرر سه‌گانه [۲۶]

همانطور که پیش‌تر ذکر شد، ما از شبکه‌ی با وزن‌های از پیش آموزش دیده برای استخراج بردارهای ویژگی از تصاویر چهره‌ها استفاده می‌کنیم و از این بردارها برای دسته‌بندی دنباله‌ها استفاده می‌کنیم. سپس از شبکه Siamese برای تنظیم دقیق‌تر شبکه عصبی و در نتیجه استخراج بردارهای ویژگی با دقت بهتر و در نتیجه دسته‌بندی با دقت بالاتر استفاده می‌کنیم و پس از تنظیم دقیق شبکه مجدد فرآیند استخراج ویژگی و دسته‌بندی دنباله‌ها را انجام می‌دهیم و انتظار داریم که دقت دسته‌بندی بهبود یابد.

به جهت این که ما مجموعه‌داده را به شکلی جمع‌آوری نمودیم که برای هر تصویر یک تصویر به عنوان سیگنا مثبت و یک تصویر به عنوان سیگنال منفی داشته باشیم، نیاز به یک تابع ضرر مانند تابع ضرر سه‌گانه داریم که نحوه‌ی محاسبه‌ی ضرر در آن به شرح رابطه‌ی زیر است:

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \text{margin}, ۰)$$

که در آن A بردار پایه، P بردار مثبت و N بردار منفی. خروجی این تابع ضرر اختلاف فاصله بردار پایه از بردار مثبت است که فاصله بردار پایه از بردار منفی از آن کسر شده و با یک مقدار لبه[1] جمع شده. که اگر

[1] Margin

این حاصل این رابطه مقداری منفی شود تابع ضرر صرفا مقدار صفر را بر می‌گرداند.

به این صورت بردارهای ویژگی دنباله‌های تصاویر یک فرد در فضای حالت مسئله به هم نزدیک‌تر شده و بردارهای ویژگی دنباله‌های تصاویر افراد مختلف در فضای حالت از هم دورتر می‌شوند و انتظار می‌رود که دقت مسئله بهبود یابد.

۹-۲-۳- نتیجه گیری

در نهایت برای هر ویدیو اطلاعات زیر خروجی داده می‌شود:

- تعداد افراد مختلفی که آن ویدیو حضور داشته‌اند
- فریم‌ها و زمان‌هایی که هر فرد در ویدیو حضور داشته
- دنباله‌های حضور افراد در فریم از زمان ورود به فریم تا زمان خروج از فریم
- تداخل‌ها و هم‌زمانی‌های دنباله‌های حضور افراد در فریم
- چهره‌ی تمامی افراد در هر ثانیه و فریم از حضورشان در ویدیو همراه با مشخص کردن تعلق آن به کدام شخصیت
- دسته‌بندی دقیق از تمامی دنباله‌ها و افراد

۳-۳- مجموعه‌داده استفاده شده

داده‌های استفاده شده در این پژوهش از یک قسمت از سریال مهمونی [۲۷] استخراج و برچسب گذاری شده‌اند که در ادامه در مورد نحوه‌ی استخراج و برچسب گذاری آن‌ها بیشتر توضیح خواهیم داد. در شکل ۱۸ می‌توانید نمونه‌ای از این مجموعه‌داده را مشاهده کنید.

شماره دنباله: ۱۲
شماره فریم:۱۴۲۳
کاراکتر: ۱
محدوده چهره: (۲۳۴,۴۷۳,۶۲۴,۵۷۴)

شماره دنباله: ۱۳
شماره فریم:۱۴۶۲
کاراکتر: ۲
محدوده چهره: (۷۳۴,۱۱۷۰,۹۹۷,۱۴۱۵)

شکل ۲۷ نمونه‌ای از داده‌های تصاویر استخراج شده و برچسب خورده

به منظور تست نهایی سیستم توسط دسته‌بند و مقایسه دقت آن با حالت قبل از تنظیم دقیق شبکه، از داده‌های برچسب خورده استفاده شد که در جدول ۲ آن را مشاهده می‌کنید. ولی چون برای آموزش مجدد و تنظیم دقیق‌تر وزن‌های شبکه تشخیص ویژگی نیاز به داده‌های برچسب خورده نداریم و بصورت خودنظاره‌گر و با استفاده از سیگنال‌های تولید شده بصورت خودکار توسط سیستم این کار انجام می‌شود برای این بخش از مجموعه داده‌ی بزرگتری استفاده می‌کنیم که توسط خود سیستم بررسی شده و در قالب سیگنال مثبت و منفی تفکیک شده و در واقع مجموعه‌داده برچسب خورده زیرمجموعه‌ای از همین مجموعه‌داده است.

به منظور تنظیم دقیق‌تر شبکه و اعتبار سنجی آن در هر مرحله از آموزش، این داده‌های برچسب نخورده به دو دسته تقسیم بندی شد که در جدول ۳ آن را مشاهده می‌کنید. باید در نظر داشت که ۳ شخصیت در ویدیو حضور دارند و خطاهای آشکارساز چهره که گاهی تصاویری جز چهره‌ی افراد را نیز استخراج می‌کند با دقت بالایی توسط دسته‌بند از تصاویر شخصیت‌ها جدا می‌کنیم و به همین جهت تعداد کل شخصیت‌ها ۴ عدد ذکر شده.

جدول ۲ تقسیم بندی داده‌های برچسب خورده

	تعداد چهره	تعداد شخصیت‌ها
آموزش	۱۰۳۷۴	۴
اعتبار سنجی و تست	۲۵۹۳	۴
کل	۱۲۹۶۷	۴

جدول ۳ تقسیم بندی داده‌های برچسب نخورده

	تعداد چهره	تعداد شخصیت‌ها
آموزش	۶۸۰۲	۴
اعتبار سنجی و تست	۲۷۲۰۶	۴
کل	۳۴۰۰۸	۴

۴-۳- پیاده‌سازی برخی تنظیمات برای بهبود عملکرد سیستم

با توجه به این که مدل اولیه دقت پایینی داشت، تصمیم گرفتیم برای بهبود دقت شبکه موارد زیر را انجام دهیم تا بتوانیم به نتایج مطلوب‌تری دست پیدا کنیم:

۱) به هم ریختن ترتیب تصاویر ورودی[1]: با توجه به این که در بسیاری از مجموعه‌داده‌های موجود تصاویر مربوط به یک دسته در کنار یکدیگر و به ترتیب قرار گرفته‌اند اگر تصاویر با همان ترتیب به عنوان ورودی به شبکه داده شوند، شبکه نمی‌تواند به خوبی همه‌ی دسته‌ها را آموزش ببیند. در نتیجه پیش از ورودی دادن به شبکه ترتیب تصاویر ورودی را به صورت تصادفی به‌هم می‌زنیم و سپس آن را در اختیار شبکه قرار می‌دهیم.

۲) افزایش تعداد داده‌های برچسب‌دار: در ابتدا یک دقیقه از ویدیوی سریال مهمونی را برچسب زدیم که

[1] Shuffle

مشـاهده شـد دقت اولیه‌ی سـیسـتم در دسـته‌بندی دنبال‌های تصـاویر شـخصـیت‌ها برابر با ۱۰۰ بود و نمی‌توانسـتیم نتیجه‌ی تنظیم دقیق شـبکه را بررسـی کنیم. به همین جهت در ادامه ۱۰ دقیقه از ویدیو را برچسب زدیم تا خطای بیشتری توسط سیستم رخ بدهد و بتوانیم بررسی دقیق‌تری داشته باشیم.

۵-۳- نتیجه گیری

در این بخش به بررسـی کارهای انجام شـده جهت رفع چالش‌های مسـئله و همچنین بهبود الگوریتم‌های مورد اسـتفاده در سـیسـتم پرداختیم و در فصـل بعد به بررسـی نتایج آزمایش و همچنین ابزارهای پیاده‌سـازی خواهیم پرداخت.

فصل ۴:
نتایج و تفسیر آن‌ها

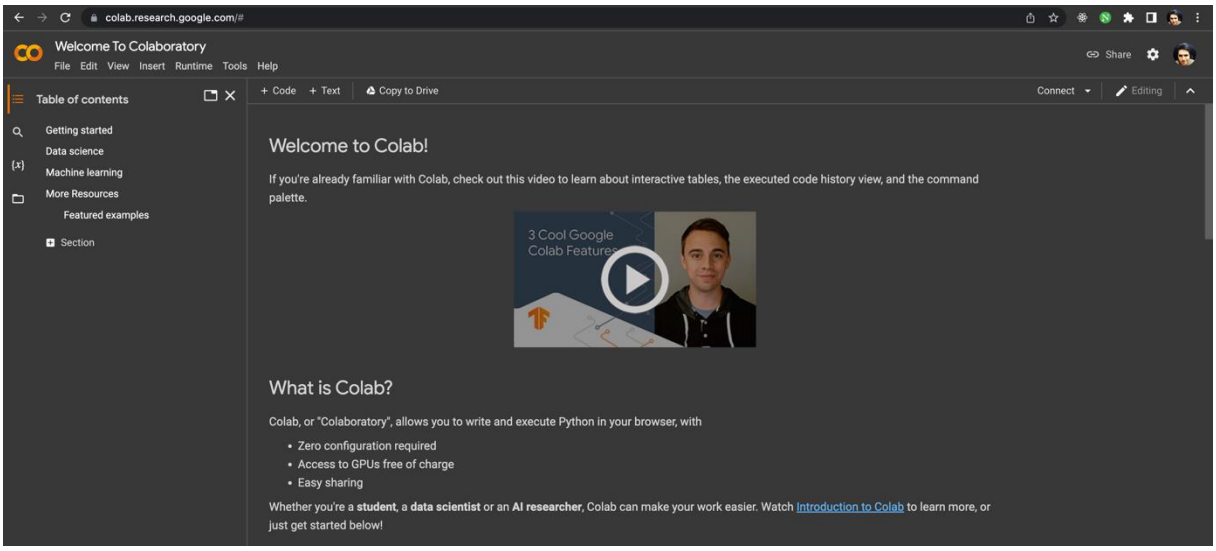
۴-۱- مقدمه

در این فصل به نحوه اجرای کدها و جزئیات آن‌ها و همچنین بررسی نتایج حاصل از آن‌ها می‌پردازیم.

۴-۲- محیط اجرای برنامه‌ها

با توجه به این که شبکه‌های عصبی معمولا بر اساس لایه‌های کانولوشنی[1] هستند و این لایه‌ها دارای محاسبات زیادی هستند، استفاده از CPU برای این شبکه‌ها در آموزش و تست، کاری مرسوم نیست. و معمولا از واحدهای گرافیکی مانند GPU استفاده می‌شود. به دلیل محدودیت اقتصادی موجود، توانایی خرید سرورهای قدرتمند گرافیکی که بتوانند سرعت مناسبی را ارائه دهند وجود نداشت و بالاجبار از سرویس‌های آماده‌ای که به رایگان در دسترس افراد قرار می‌گیرند استفاده شد.

گوگل کولب[2] یکی از این سرویس‌ها است. این سرویس با در اختیار داشتن یک حساب کاربری گوگل، به شما این امکان را می‌دهد که فایل‌های خود را که به زبان پایتون[3] نوشته شده در قالب ژوپیتر نوت‌بوک[4]، به مدت ۱۲ ساعت به صورت مداوم اجرا کنید. این سرویس سه مدل اجرایی دارد که شامل CPU، GPU و TPU است. مدل اجرایی TPU، یک سخت‌افزار توسعه‌یافته برای محاسبات هوش مصنوعی است که در حال حاضر در نسخه‌ی آزمایشی قرار دارد.



شکل ۲۸ سامانه Google Colab

یکی از مشکلاتی که در استفاده از سرویس گوگل کولب وجود داشت این بود که اگر برای لحظه‌ای ارتباط اینترنتی دستگاه قطع شود، فرآیند یادگیری شبکه متوقف می‌شود و بایستی یادگیری از ابتدا آغاز

[1] convolutioal
[2] Google colab
[3] python
[4] jupyter notebook

شود. با توجه به بزرگ بودن حجم مجموعه‌داده‌های استفاده شده و همچنین حجم بالای محاسبات لازم برای آموزش شبکه، فرآیند یادگیری شبکه مدت زمان زیادی را لازم داشت. متاسفانه در اکثر تلاش‌هایی که برای استفاده از گوگل کولب برای آموزش شبکه‌ها داشتیم، پس از گذشت زمان نسبتا کوتاهی اتصال به سرورهای گوگل با مشکل مواجه می‌شد و این باعث می‌شد که فرآیند آموزش شبکه به انتها نرسد.

خوشبختانه با راهنمایی‌های استاد راهنما در طول زمان انجام پروژه، توابعی برای بخش‌های مختلف سیستم پیاده سازی کردیم که در هر مرحله نتایج آن بخش و بهترین مدل را و همچنین داده‌های مربوط به نتایج آموزش هر مرحله را ذخیره کنیم و با این ترفند بتوانیم در صورت قطع شدن از کولب[1] فرآیند یادگیری را ادامه دهیم . و همچنین برای کمتر کردن زمان تکرار اجرا در فرآیند توسعه نیز این توابع به کمک ما آمدند. در شکل ۲۸ نمایی از محیط سامانه Google Colab را مشاهده می‌کنید.

۳-۴- پیاده‌سازی شبکه

در این قسمت به ترتیب به پیاده سازی بخش‌های

- سیستم ردیابی ویدیو و آشکارساز چهره
- بخش شبکه عصبی
- بخش محاسبه بردارهای ویژگی دنباله‌ها و آماده‌سازی ورودی‌های دسته‌بند و دسته‌بندی
- بخش ورودی‌های شبکه
- بخش شبکه Siamese و تابع ضرر سه‌گانه

[1] Colab

۱-۳-۴- پیاده سازی سیستم ردیابی ویدیو و آشکارساز چهره

در این بخش ابتدا به بررسی کد پیاده‌سازی شده برای بخش آشکارساز چهره می‌پردازیم:

```
detector = build_detector('RetinaNetMobileNetV1',
        confidence_threshold = .5, nms_iou_threshold = .3,
        max_resolution = 250)

class Face():
    """
    Create an object and update an attribute when it is ready.

    constructor params:
        - box: a list like : [x1, y1, x2, y2]
        - confidence: if detector return probability of detected
face, put that here.
        - lmarks: put landmarks of face here, 5 or 68 or whatever
        - features: output of FaceRepresentor
        - id: real identity of face
        - pred_id: predicted identity of face

    """
    def __init__(self, box=None, detection_confidence=None,\
                 landmarks=None, embedding=None, id=None):
        self.box = box
        self.confidence = detection_confidence
        self.lmarks = landmarks
        self.id = id
        self.crop = None
        self.features = embedding
        self.pred_id = 0


    def __str__(self):
        attrs = vars(self)
        return ', '.join("%s: %s" % item for item in attrs.items())
```

```
def face_detection(image):
    '''
    Face Detection

    Parameters:
        image (numpy.ndarray): The input image.

    Returns:
        faces (list): The list of faces in the input image.
    '''

    bbox, lmarks = detector.batched_detect_with_landmarks(
      np.expand_dims(image, axis = 0))
    faces = []
    for i, box in enumerate(bbox[0]):
        f = Face()
        [x1, y1, x2, y2, pr] = box.astype(int)
        f.box = [x1, y1, x2, y2]
        f.lmarks = lmarks[0][i].astype(int)
        faces.append(f)
    return faces
```

همانطور که مشاهده می‌کنید در پیاده سازی بخش آشکارساز چهره از آشکارساز چهره RetinaFace با رزولشن تصویر ۲۵۰ پیکسل استفاده شده و با فراخوانی تابع face_detection() بر روی هر تصویر تمامی چهره‌های درون آن شناسایی شده و در قالب یک لیست که در آن به ازای هر چهره شناسایی شده یک باکس چهره به همراه نقاط عطف چهره وجود دارد برگشت داده می‌شود.

در این بخش به بررسی کد پیاده‌سازی شده برای بخش ردیابی چهره‌های ویدیو می‌پردازیم:

```
class CentroidTracker():
    def __init__(self, maxDisappeared=0):
        # initialize the next unique object ID along with two
ordered
        # dictionaries used to keep track of mapping a given object
        # ID to its centroid and number of consecutive frames it has
        # been marked as "disappeared", respectively
        self.nextObjectID = 0
        self.objects = OrderedDict()
        self.disappeared = OrderedDict()
        # store the number of maximum consecutive frames a given
        # object is allowed to be marked as "disappeared" until we
        # need to deregister the object from tracking
        self.maxDisappeared = maxDisappeared

    def register(self, centroid):
        # when registering an object we use the next available
object
        # ID to store the centroid
        self.objects[self.nextObjectID] = centroid
        self.disappeared[self.nextObjectID] = 0
        self.nextObjectID += 1

    def deregister(self, objectID):
        # to deregister an object ID we delete the object ID from
        # both of our respective dictionaries
        del self.objects[objectID]
        del self.disappeared[objectID]

    def update(self, rects):
        # check to see if the list of input bounding box rectangles
        # is empty
        if len(rects) == 0:
            # loop over any existing tracked objects and mark them
            # as disappeared
            for objectID in list(self.disappeared.keys()):
                self.disappeared[objectID] += 1
                # if we have reached a maximum number of consecutive
                # frames where a given object has been marked as
                # missing, deregister it
                if self.disappeared[objectID] > self.maxDisappeared:
                    self.deregister(objectID)
            # return early as there are no centroids or tracking
info
            # to update
            return self.objects
```

```
        # initialize an array of input centroids for the current
frame
        inputCentroids = np.zeros((len(rects), 2), dtype="int")
        # loop over the bounding box rectangles
        for (i, (startX, startY, endX, endY)) in enumerate(rects):
            # use the bounding box coordinates to derive the
centroid
            cX = int((startX + endX) / 2.0)
            cY = int((startY + endY) / 2.0)
            inputCentroids[i] = (cX, cY)
        # if we are currently not tracking any objects take the
input
        # centroids and register each of them
        if len(self.objects) == 0:
            for i in range(0, len(inputCentroids)):
                self.register(inputCentroids[i])
        # otherwise, are are currently tracking objects so we need
to
        # try to match the input centroids to existing object
        # centroids
        else:
            # grab the set of object IDs and corresponding centroids
            objectIDs = list(self.objects.keys())
            objectCentroids = list(self.objects.values())
            # compute the distance between each pair of object
            # centroids and input centroids, respectively -- our
            # goal will be to match an input centroid to an existing
            # object centroid
            D = dist.cdist(np.array(objectCentroids),
inputCentroids)
            # in order to perform this matching we must (1) find the
            # smallest value in each row and then (2) sort the row
            # indexes based on their minimum values so that the row
            # with the smallest value is at the *front* of the index
            # list
            rows = D.min(axis=1).argsort()
            # next, we perform a similar process on the columns by
            # finding the smallest value in each column and then
            # sorting using the previously computed row index list
            cols = D.argmin(axis=1)[rows]
            # in order to determine if we need to update, register,
            # or deregister an object we need to keep track of which
            # of the rows and column indexes we have already
examined
            usedRows = set()
            usedCols = set()
```

```
            # loop over the combination of the (row, column) index
            # tuples
            for (row, col) in zip(rows, cols):
                # if we have already examined either the row or
                # column value before, ignore it
                # val
                if row in usedRows or col in usedCols:
                    continue
                # otherwise, grab the object ID for the current row,
                # set its new centroid, and reset the disappeared
                # counter
                objectID = objectIDs[row]
                self.objects[objectID] = inputCentroids[col]
                self.disappeared[objectID] = 0
                # indicate that we have examined each of the row and
                # column indexes, respectively
                usedRows.add(row)
                usedCols.add(col)
            # compute both the row and column index we have NOT yet
            # examined
            unusedRows = set(range(0,
D.shape[0])).difference(usedRows)
            unusedCols = set(range(0,
D.shape[1])).difference(usedCols)
            # in the event that the number of object centroids is
            # equal or greater than the number of input centroids
            # we need to check and see if some of these objects have
            # potentially disappeared
            if D.shape[0] >= D.shape[1]:
                # loop over the unused row indexes
                for row in unusedRows:
                    # grab the object ID for the corresponding row
                    # index and increment the disappeared counter
                    objectID = objectIDs[row]
                    self.disappeared[objectID] += 1
                    # check to see if the number of consecutive
                    # frames the object has been marked
"disappeared"
                    # for warrants deregistering the object
                    if self.disappeared[objectID] >
self.maxDisappeared:
                        self.deregister(objectID)
            # otherwise, if the number of input centroids is greater
            # than the number of existing object centroids we need
to
            # register each new input centroid as a trackable object
```

```
            else:
                for col in unusedCols:
                    self.register(inputCentroids[col])
        # return the set of trackable objects
        return self.objects
```

```
scene_start_frame = [0]
def sequence_shot_detect(filename):
    scene_list = detect(filename, ContentDetector())
    for scene in scene_list:
        # print(scene[1].frame_num)
        scene_start_frame.append(scene[1].frame_num)
    print("Shot detection finished! shots detected:",
len(scene_start_frame))
```

```
frame_boxes = []
def sequence_save(filename):
    # initialize our centroid tracker and frame dimensions
    ct = CentroidTracker()
    (H, W) = (None, None)
    print("[INFO] starting video stream...")
    cap = cv2.VideoCapture(filename)
    fps = cap.get(cv2.CAP_PROP_FPS) # Gets the frames per second
    print('video fps is: ', fps)

    frameID = 0
    sequencesIDs = []

    # Shot Detect params:
    scene_start_frame_pointer = 1
    fnum = scene_start_frame[scene_start_frame_pointer]
    max_pointer = len(scene_start_frame) - 1

    # loop over the frames from the video stream
    while True:
        # read the next frame from the video stream and resize it
        # frame = vs.read()
        ret, frame = cap.read()
        if (not ret):
            break
```

```
        frame = imutils.resize(frame, width=400)
        # if the frame dimensions are None, grab them
        if W is None or H is None:
            (H, W) = frame.shape[:2]
        rects = []
        if (frameID == fnum):
            if (scene_start_frame_pointer == max_pointer):
                frameID += 1
                faces = []
                objects = ct.update(rects)
                continue
            else:
                scene_start_frame_pointer += 1
                fnum = scene_start_frame[scene_start_frame_pointer]
                frameID += 1
                faces = []
                objects = ct.update(rects)
                continue
        faces = face_detection(frame)
        # loop over the detections
        for i in range(len(faces)):
            x, y, w, h = faces[i].box
            box = np.array([x, y, w, h])
            rects.append(box.astype("int"))
            (startX, startY, endX, endY) = box.astype("int")
        # update our centroid tracker using the computed set of
        # bounding box rectangles
        objects = ct.update(rects)
        # loop over the tracked objects
        in_frame_ctr = 0
        for (objectID, centroid) in objects.items():
            # draw both the ID of the object and the centroid of the
            # object on the output frame
            try:
                box = rects[in_frame_ctr]
            except:
                print("LOST FACE")

            x, y, w, h = box
```

```
            if x < 0:
                x = 0
            if y < 0:
                y = 0
            if w < 0:
                w = 0
            if h < 0:
                h = 0

            if (not objectID in sequencesIDs):
                sequencesIDs.append(objectID)
cv2.imwrite("/content/sequences/sequence%d_frame%d.jpg" %(objectID,
frameID), frame[int(y):int(h), int(x):int(w)])
            else:
                try:
cv2.imwrite("/content/sequences/sequence%d_frame%d.jpg" %(objectID,
frameID), frame[int(y):int(h), int(x):int(w)])
                except:
                    print("FOUND AN ERROR!")
            d = [{
                "fileName": "sequence%d_frame%d.jpg" %(objectID,
frameID),
                "objectID": objectID,
                "frameID": frameID,
                "bbox": (int(y), int(h), int(x), int(w))
            }]
            frame_boxes.extend(d)
            in_frame_ctr += 1
        frameID += 1
    # do a bit of cleanup
    cv2.destroyAllWindows()
    # cap.stop()
    print("DONE!")
```

همانطور که مشاهده می‌کنید در پیاده سازی بخش ردیاب چهره‌های ویدیو از یک سیستم ردیابی اشیاء در ویدیوی مبتنی بر محدوده چهره و بر اساس یافتن مرکز محدوده چهره استفاده می‌کنیم و در ردیابی چهره‌های موجود در ویدیو ابتدا از یک ماژول تشخیص تغییر صحنه استفاده می‌کنیم و ویدیو را به چند ویدیو تقسیم می‌کنیم که در هر ویدیو صحنه ثابت است و به این طریق سیستم را در برابر تغییرات مربوط به تغییر زاویه‌ی دوربین و تغییرات صحنه تقویت می‌کنیم و سپس روی هر کدام از این ویدیوها به ترتیب فریم به فریم می‌خوانیم و چهره‌های آن فریم را استخراج می‌کنیم و دنباله‌های چهره را به دست می‌آوریم.

۲-۳-۴- پیاده سازی بخش شبکه عصبی محاسبه بردارهای ویژگی

در این بخش به بررسی کد پیاده سازی شده برای ساخت شبکه عصبی و محاسبه بردارهای ویژگی از روی تصاویر چهره‌ی افراد می‌پردازیم:

```
# create a vggface model
base_cnn = VGGFace(model='resnet50', include_top=False,
input_shape=target_shape + (3,), pooling='avg')

embedding = Model(base_cnn.input, base_cnn.output, name="Embedding")

trainable = False
for layer in base_cnn.layers:
    if layer.name == "conv5_3_1x1_increase":
        trainable = True
    layer.trainable = trainable
```

```
def preprocess_image(filename):
    """
    Load the specified file as a JPEG image, preprocess it and
    resize it to the target shape.
    """
    image_string = tf.io.read_file(filename)
    image = tf.image.decode_jpeg(image_string, channels=3)
    image = tf.image.convert_image_dtype(image, tf.float32)
    image = tf.image.resize(image, target_shape)
    return image
def preprocess_triplets(anchor, positive, negative):
    """
    Given the filenames corresponding to the three images, load and
    preprocess them.
    """
    return (
        preprocess_image(anchor),
        preprocess_image(positive),
        preprocess_image(negative),
    )
```

```
def get_sequence_embeddings(filelist):
    seq_emb_data = []
    faces_result = []
    for f in filelist:
        pixels = cv2.imread(f, 1)
        # resize pixels to the model size
        image = Image.fromarray(pixels)
        image = image.resize((200, 200))
        face_array = asarray(image)
        extracted_face = asarray(face_array, 'float32')
        # prepare the face for the model, e.g. center pixels
        extracted_face = resnet.preprocess_input(extracted_face)
        extracted_face = np.expand_dims(extracted_face, axis=0)
        emm = embedding(extracted_face)[0]
        objectID = f.split('/')[-
1].split('.')[0].split('_')[0].split('sequence')[1]
        frameID = f.split('/')[-
1].split('.')[0].split('_')[1].split('frame')[1]

        d = [{
                "imagePath": f,
                "objectID": objectID,
                "frameID": frameID,
                "encoding": emm
            }]
        seq_emb_data.extend(d)
    return seq_emb_data
```

همانطور که مشاهده می‌کنید در این بخش ابتدا شبکه عصبی با شبکه‌ی پایه‌ی ResNet۵۰ با وزن‌های vggface ساخته شده و سپس چند تابع کمکی برای پیش پردازش ورودی‌های شبکه تعریف شده و در انتها یک تابع با نام get_sequence_embeddings() تعریف شده که با صدا زدن آن و دادن لیست کل تصاویر استخراج شده توسط سیستم ردیاب بردارهای ویژگی آن‌ها را محاسبه می‌کند و برای هر تصویر یک دیکشنری حاوی آدرس فایل چهره و شماره دنباله‌ی آن و شماره فریم آن و بردار ویژگی آن را بر می‌گرداند.

۳-۳-۴- پیاده سازی بخش محاسبه بردارهای ویژگی دنباله‌ها و آماده‌سازی ورودی‌های دسته‌بند

در این بخش به بررسی کد پیاده سازی شده برای محاسبه بردارهای ویژگی هر دنباله و آماده‌سازی

دنباله‌ها جهت دسته‌بندی می‌پردازیم:

```
seq_emb_data = sorted(seq_emb_data, key=lambda x:int(x['objectID']))
seq_emb_data_dict = {}
i = 0
for item in seq_emb_data:
    i += 1
    imagePath = item["imagePath"]
    objectID = item["objectID"]
    if objectID in seq_emb_data_dict:
        d = [item]
        seq_emb_data_dict[objectID].extend(d)
    else:
        seq_emb_data_dict[objectID] = [item]
seq_emb_data_objectIDs = seq_emb_data_dict
seq_emb_data_objectIDs_avg = []
for item in seq_emb_data_objectIDs:
    embs = []
    sum_embs = 0
    for idx in range(len(seq_emb_data_objectIDs[item])):
        embs.append(seq_emb_data_objectIDs[item][idx])
        sum_embs += (seq_emb_data_objectIDs[item][idx]["encoding"])
    d = [{
            "objectID": seq_emb_data_objectIDs[item][0]['objectID'],
            "encoding_avg":
(sum_embs/len(seq_emb_data_objectIDs[item])),
            "list_of_objects": embs
    }]
    seq_emb_data_objectIDs_avg.extend(d)

seq_emb_data_objectIDs_avg = np.array(seq_emb_data_objectIDs_avg)
seq_emb_data_objectIDs_avg_encodings = [d["encoding_avg"] for d in
seq_emb_data_objectIDs_avg]
```

```
# cluster the embeddings
print("[INFO] clustering...")
print(len(seq_emb_data_objectIDs_avg_encodings[0]))
clt = AgglomerativeClustering(n_clusters = 4)
clt.fit(seq_emb_data_objectIDs_avg_encodings)
labelIDs = np.unique(clt.labels_)
numUniqueFaces = len(np.where(labelIDs > -1)[0])
print("[INFO] # unique faces: {}".format(numUniqueFaces))
```

```
import pandas as pd
col_list = ["seq_no", "char_no", "other_chars"]
df =
pd.read_csv('/content/drive/MyDrive/RetinaFiles/seq_retina_mehmuni10
.csv', usecols=col_list)
labels_true = []
for i in (df["char_no"]):
    labels_true.append(i)

from sklearn.metrics.cluster import normalized_mutual_info_score
labels_pred = clt.labels_
print("Before Fine-Tuning NMI:")
normalized_mutual_info_score(labels_true, labels_pred)
```

همانطور که مشـاهده می‌کنید در این بخش ابتدا با میانگین گرفتن روی بردارهای ویژگی تمام چهره‌های موجود در هر دنباله، بردار ویژگی میانگین را برای هر دنباله محاسبه می‌کنیم و سپس با استفاده از دسته‌بند سلسله مراتبی تجمعی دنباله‌ها را دسته‌بندی می‌کنیم. قابل ذکر است که مجموعه‌داده‌ای که برچسب زده‌ایم مجموعه‌داده‌ای است که توسط سیستم آشکار ساز چهره و سیستم ردیاب چهره از ویدیو استخراج شده است بنابراین روند طی شده تفاوتی با زمانی که کل سیستم بصورت خود-نظاره‌گر فعالیت می‌کند ندارد. در ادامه برچسب‌های اصلی دنباله‌ها را از فایل csv که داریم می‌خوانیم و با استفاده از معیار[1] NMI برچسب‌های پیش‌بینی شده برای دنباله‌ها توسط دسته‌بند را با برچسب‌های اصلی دنباله‌ها مقایسه می‌کنیم تا دقت دسته‌بند روی مجموعه داده‌ی خود را بسنجیم.

۴-۳-۴- پیاده سازی بخش ورودی‌های شبکه

در این بخش به بررسی کد پیاده سازی شده برای آماده‌سازی ورودی‌های سه‌گانه جهت ورودی دادن به شبکه Siamese و تنظیم دقیق‌تر شبکه می‌پردازیم:

[1] Metric

```
seq_emb_data = sorted(seq_emb_data, key=lambda x:int(x['objectID']))
seq_emb_data_frameID_dict = {}
i = 0
for item in seq_emb_data:
    i += 1
    imagePath = item["imagePath"]
    frameID = item["frameID"]
    if frameID in seq_emb_data_frameID_dict:
        d = [item]
        seq_emb_data_frameID_dict[frameID].extend(d)
    else:
        seq_emb_data_frameID_dict[frameID] = [item]

seq_emb_data_dict = {}
i = 0
for item in seq_emb_data:
    i += 1
    imagePath = item["imagePath"]
    objectID = item["objectID"]
    if objectID in seq_emb_data_dict:
        d = [item]
        seq_emb_data_dict[objectID].extend(d)
    else:
        seq_emb_data_dict[objectID] = [item]
seq_data_objectIDs_frameIDs = {}

for i in range(len(seq_emb_data_objectIDs)):
    objectID = seq_emb_data_objectIDs[str(i)][0]["objectID"]
    seq_data_objectIDs_frameIDs[objectID] = []
    for j in range(len(seq_emb_data_objectIDs[str(i)])):
        d = [seq_emb_data_objectIDs[str(i)][j]["frameID"]]
        seq_data_objectIDs_frameIDs[objectID].extend(d)

def intersection_of_lists(lst1, lst2):
    lst3 = [value for value in lst1 if value in lst2]
    return lst3
```

```
sequences_co_occurrences = {}

for i in range(len(seq_data_objectIDs_frameIDs)):
    objectID = str(i)
    sequences_co_occurrences[objectID] = {
        "frameIDs" : seq_data_objectIDs_frameIDs[objectID],
        "cooccurrences": []
    }
    for j in range(len(seq_data_objectIDs_frameIDs)):
        second_objectID = str(j)
        if i == j:
            continue
        else:
            intersection_result = intersection_of_lists(

seq_data_objectIDs_frameIDs[objectID],

seq_data_objectIDs_frameIDs[second_objectID],
                                    )
            if (len(intersection_result) != 0):
                d = [second_objectID]

      sequences_co_occurrences[objectID]["cooccurrences"].extend(d)
```

همانطور که مشاهده می‌کنید، ابتدا داده‌ها را بر اساس شماره فریم و شماره دنباله دسته‌بندی می‌کنیم تا بتوانیم جهت ساخت ورودی‌های مناسب شبکه در آن‌ها جستوجو انجام بدهیم و با استفاده از همین دسته‌بندی‌ها و تابع کمکی اشتراک دو لیست[1] همزمانی‌های رخداد دنباله‌ها[2] را نیز می‌یابیم و از این دسته‌بندی‌ها و داده‌های مربوط به همزمانی رخداد دنبلله‌ها داده‌ها را به شکل زیر جهت ورودی دادن به شبکه Siamese آماده می‌کنیم:

[1] List intersection
[2] Co-occurrences of sequences

```
seqq1 = glob.glob('/content/sequences/*')
seqq1.sort()
frames_recovered_from_sequences = []
for i in seqq1:
    frames_recovered_from_sequences.extend(glob.glob(i + '/*'))
print(len(frames_recovered_from_sequences))
print(frames_recovered_from_sequences[0])
anchor_images = sorted(frames_recovered_from_sequences)
print(frames_recovered_from_sequences[0])
print("ANCHORS GATHERED!")
```

همانطور که مشـــاهده می‌کنید، ابتدا همه‌ی داده‌ها را در لیســـت لنگر[1] می‌چینیم که پایه‌ی ورودی سه‌گانه به شبکه Siamese ما است و در آن هر چهره‌ی استخراج شده توسط آشکارساز چهره یک‌بار قرار می‌گیرد.

```
pos_list = []
for anchor in anchor_images:
    sample = anchor.split('/')[-1].split('.')[0].split('_')
    objectID = sample[0].split('sequence')[-1]
    frameID = sample[1].split('frame')[-1]
    objectID_query = seq_emb_data_objectIDs[str(objectID)]
    random_sample = random.randint(0, (len(objectID_query)-1))
    pos_list.append(objectID_query[random_sample]["imagePath"])
```

در ادامه برای هر تصویر لنگر باید یک تصویر به‌عنوان جفت مثبت قرار بدهیم و همانطور که مشاهده می‌کنید منطق ما در انتخاب این تصـویر، انتخاب یک تصـویر به صـورت تصـادفی از همان دنباله‌ای اسـت که تصـویر پایه به آن تعلق دارد. بدین صـورت این تصـویر یک سـیگنال مثبت قطعی برای تصـویر لنگر اسـت، چرا که تصویری از همان شخص می‌باشد.

[1] Anchor

```
neg_list = []
for anchor in anchor_images:
    sample = anchor.split('/')[-1].split('.')[0].split('_')
    objectID = sample[0].split('sequence')[-1]
    frameID = sample[1].split('frame')[-1]
    objectID_co_occurrences =
sequences_co_occurrences[str(objectID)]["cooccurrences"]

    if (len(objectID_co_occurrences) > 0):
        objectID_co_occurred =
objectID_co_occurrences[random.randint(0,
(len(objectID_co_occurrences)-1))]
        objectID_query =
seq_emb_data_objectIDs[str(objectID_co_occurred)]
        random_sample = random.randint(0, (len(objectID_query)-1))
        neg_list.append(objectID_query[random_sample]["imagePath"])
    else:
        cluster_of_anchor = clustering_objectIDs[str(objectID)]
        target_objectIDs = []
        for i in range(len(clustering_objectIDs)):
            if clustering_objectIDs[str(i)] != cluster_of_anchor:
                target_objectIDs.append(str(i))

        if (len(target_objectIDs) > 0):
            random_sample = random.randint(0,
(len(target_objectIDs)-1))
            sample_objectID = target_objectIDs[random_sample]
            objectID_query =
seq_emb_data_objectIDs[str(sample_objectID)]
            random_sample = random.randint(0, (len(objectID_query)-
1))

neg_list.append(objectID_query[random_sample]["imagePath"])
        else:
            print("FOUND NO MATCH!!!")
```

در ادامه برای هر تصـــویر لنگر باید یک تصـــویر هم به‌عنوان جفت منفی قرار بدهیم و همانطور که مشاهده می‌کنید منطق ما در انتخاب این تصویر، به این شکل است که برای هر تصویر لنگر مراحل زیر را به ترتیب جهت پیدا کردن جفت منفی مناسب برای آن طی کنیم:

- ابتدا بررســی می‌کنیم ببینیم آیا این دنباله با دنباله دیگری هم‌زمانی رخداد داشــته یا نه. اگر این دنباله با دنباله‌ی دیگری هم‌زمانی رخداد داشته باشد یعنی این دو دنباله در حداقل یک فریم هر

دو در تصـویر باهم حاضـر بوده‌اند و این بدین معنی اسـت که این دو دنباله قطعا متعلق به دو نفر مختلف می‌باشند و این یک سیگنال منفی قطعی است.

- در صورت نداشـتن دنباله‌ی هم‌زمان، به این دلیل که این سیسـتم یک سیسـتم خودنظاره‌گر است و نمی‌توانیم از اطلاعاتی که به صـورت دسـتی در برچسـب گذاری افزوده‌ایم اسـتفاده کنیم، از اطلاعات مربوط به دسته‌بند که در بخش قبلی بدست آوردیم استفاده می‌کنیم و برای آن تصویر لنگر، بصورت تصادفی، یک تصویر از دنباله‌ای که در دسته‌ای متفاوت از تصویر لنگر عضو است را انتخاب می‌کنیم و این سیگنال نیز با دقت بسیار بالایی سیگنال منفی صحیحی است.

```
anchor_images = anchor_images

positive_images = pos_list

negative_images = neg_list

image_count = len(anchor_images)

anchor_dataset = tf.data.Dataset.from_tensor_slices(anchor_images)
positive_dataset =
tf.data.Dataset.from_tensor_slices(positive_images)
negative_dataset =
tf.data.Dataset.from_tensor_slices(negative_images)
```

```
dataset = tf.data.Dataset.zip((anchor_dataset, positive_dataset,
negative_dataset))
dataset = dataset.shuffle(buffer_size=1024)
dataset = dataset.map(preprocess_triplets)

train_dataset = dataset.take(round(image_count * 0.8))
val_dataset = dataset.skip(round(image_count * 0.8))

train_dataset = train_dataset.batch(32, drop_remainder=False)
train_dataset = train_dataset.prefetch(8)

val_dataset = val_dataset.batch(32, drop_remainder=False)
val_dataset = val_dataset.prefetch(8)
```

در انتها هر لیســت تصـاویر لنگر، جفت‌های مثبت و جفت‌های منفی را بارگذاری می‌کنیم و مجموعه‌داده[۱] مورد نیاز شـــبکه Siamese را از روی آن‌ها می‌ســـازیم، مجموعه داده را به هم مخلوط می‌کنیم[۲] تا در برابر ترتیب داده‌ها تاثیر ناپذیر باشد و در نهایت با نسبت ۸ به ۲ از ۸۰ درصد داده‌ها برای یادگیری و از ۲۰ درصد داده‌ها برای اعتبارسنجی استفاده می‌کنیم.

```
def visualize(anchor, positive, negative):
    """Visualize a few triplets from the supplied batches."""

    def show(ax, image):
        ax.imshow(image)
        ax.get_xaxis().set_visible(False)
        ax.get_yaxis().set_visible(False)

    fig = plt.figure(figsize=(9, 9))

    axs = fig.subplots(3, 3)
    for i in range(3):
        show(axs[i, 0], anchor[i])
        show(axs[i, 1], positive[i])
        show(axs[i, 2], negative[i])
```

```
visualize(*list(train_dataset.take(1).as_numpy_iterator())[0])
```

[1] Dataset
[2] Shuffle

شکل ۲۹ سه نمونه ورودی شبک

همانطور که در شـــکل ۲۹ مشـــاهده می‌کنید، داده‌ها با اســـتفاده از این تکنیک‌ها با دقت بســـیار خوبی ســـیگنال‌های مثبت و منفی مورد نیاز شـــبکه را برای هر تصـــویر لنگر فراهم می‌کنند و تمام این فرآیندها بصورت خودنظاره‌گر انجام می‌شود و از داده‌های برچسب‌گذاری شده فقط در سنجش دقت کلی سیستم استفاده شده است.

۵-۳-۴- بخش شبکه Siamese و تابع ضرر سه‌گانه

در این بخش به بررسی کد پیاده سازی شده برای محاسبه فاصله‌ی بین بردارهای ویژگی سیگنال‌های مثبت و منفی با تصویر لنگر و استفاده از آن در جهت تنظیم دقیق‌تر شبکه عصبی استخراج بردار ویژگی با استفاده از طراحی یک شبکه Siamese با تابع ضرر سه‌گانه می‌پردازیم:

```
class DistanceLayer(layers.Layer):
    """
    This layer is responsible for computing the distance between the
anchor
    embedding and the positive embedding, and the anchor embedding
and the
    negative embedding.
    """
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def call(self, anchor, positive, negative):
        ap_distance = tf.reduce_sum(tf.square(anchor - positive), -
1)
        an_distance = tf.reduce_sum(tf.square(anchor - negative), -
1)
        return (ap_distance, an_distance)
```

```
anchor_input = layers.Input(name="anchor", shape=target_shape +
(3,))
positive_input = layers.Input(name="positive", shape=target_shape +
(3,))
negative_input = layers.Input(name="negative", shape=target_shape +
(3,))

distances = DistanceLayer()(
    embedding(resnet.preprocess_input(anchor_input)),
    embedding(resnet.preprocess_input(positive_input)),
    embedding(resnet.preprocess_input(negative_input)),
)

siamese_network = Model(
    inputs=[anchor_input, positive_input, negative_input],
outputs=distances
)
```

همانطور که مشـــاهده می‌کنید، ابتدا یک لایه محاســـبه‌ی فاصـــله طراحی می‌کنیم که این لایه مســـئول محاسـبه‌ی فاصـله‌ی بین بردار ویژگی تصویر لنگر و جفت‌های مثبت و منفی اسـت و این دو مقدار را بصـورت یک دوتایی‌مرتب[1] خروجی می‌دهد. در ادامه شـــبکه‌ی Siamese را با اضـــافه کردن لایه‌ی فاصـــله در بالای

[1] Tuple

شبکه‌ی عصبی‌ای که داشتیم بصورتی طراحی می‌کنیم که ورودی شبکه Siamese ورودی‌های سه‌گانه‌ای که چیدیم باشد و خروجی آن لایه‌ی فاصله‌ای که طراحی کردیم باشد.

```
class SiameseModel(Model):
    """The Siamese Network model with a custom training and testing
loops.

    Computes the triplet loss using the three embeddings produced by
the
    Siamese Network.

    The triplet loss is defined as:
       L(A, P, N) = max(‖f(A) - f(P)‖² - ‖f(A) - f(N)‖² + margin, 0)
    """
    def __init__(self, siamese_network, margin=0.5):
        super(SiameseModel, self).__init__()
        self.siamese_network = siamese_network
        self.margin = margin
        self.loss_tracker = metrics.Mean(name="loss")

    def call(self, inputs):
        return self.siamese_network(inputs)

    def train_step(self, data):
        # GradientTape is a context manager that records every
operation that
        # you do inside. We are using it here to compute the loss so
we can get
        # the gradients and apply them using the optimizer specified
in
        # `compile()`.
        with tf.GradientTape() as tape:
            loss = self._compute_loss(data)

        # Storing the gradients of the loss function with respect to
the
        # weights/parameters.
        gradients = tape.gradient(loss,
self.siamese_network.trainable_weights)

        # Applying the gradients on the model using the specified
optimizer
        self.optimizer.apply_gradients(
            zip(gradients, self.siamese_network.trainable_weights)
        )
        # Let's update and return the training loss metric.
        self.loss_tracker.update_state(loss)
        return {"loss": self.loss_tracker.result()}
```

```
    def test_step(self, data):
        loss = self._compute_loss(data)

        # Let's update and return the loss metric.
        self.loss_tracker.update_state(loss)
        return {"loss": self.loss_tracker.result()}

    def _compute_loss(self, data):
        # The output of the network is a tuple containing the
distances
        # between the anchor and the positive example, and the
anchor and
        # the negative example.
        ap_distance, an_distance = self.siamese_network(data)

        # Computing the Triplet Loss by subtracting both distances
and
        # making sure we don't get a negative value.
        loss = ap_distance - an_distance
        loss = tf.maximum(loss + self.margin, 0.0)
        return loss

    @property
    def metrics(self):
        # We need to list our metrics here so the `reset_states()`
can be
        # called automatically.
        return [self.loss_tracker]
```

همانطور که مشاهده می‌کنید، در ادامه مدل شبکه Siamese را با حلقه‌های یادگیری و اعتبارسنجی سفارشی به شکلی طراحی می‌کنیم که بتوانیم شبکه‌ی Siamese ای که تعریف کردیم را آموزش دهیم و خروجی بردارهای ویژگی سه تصویر ورودی آن و فاصله‌های آن‌ها را به تابع ضرر سه‌گانه بدهیم و بدین ترتیب شکبه Siamese را آموزش بدهیم و شبکه عصبی استخراج بردارهای ویژگی را تنظیم دقیق‌تر[1] کنیم.

[1] Fine-tune

۴-۴- پیاده‌سازی برخی تنظیمات برای بهبود عملکرد سیستم

۴-۴-۱- به هم ریختن تصاویر ورودی

با استفاده از قطعه کد زیر ترتیب تصاویر لنگر ورودی و جفت‌های مثبت و منفی آن‌ها را به صورت تصادفی به هم می‌ریزیم.

```
dataset = tf.data.Dataset.zip((anchor_dataset, positive_dataset,
negative_dataset))
dataset = dataset.shuffle(buffer_size=1024)
dataset = dataset.map(preprocess_triplets)
```

این قطعه کد ابتدا لنگر و جفت‌ها را باهم بصورت سه‌تایی مرتب‌هایی کنار هم قرار می‌دهد و سپس سه‌تایی مرتب‌ها را به هم می‌ریزد و در واقع در این فرآیند صحت سیگنال‌ها از بین نمی‌رود. این قطعه کد در بخش آماده‌سازی ورودی‌ها پیاده سازی شده است.

۴-۴-۲- تراز کردن تصاویر چهره

این قطعه کد، تصاویر چهره‌های استخراج شده را تراز می‌کند و به این ترتیب تصاویر چهره‌های استخراج شده همگی از نظر تراز چهره با هم یکسان هستند و تفاوت ویژگی‌های استخراج شده از آن‌ها بیشتر معطوف به دیگر ویژگی‌های آن‌ها می‌شود.

```
def face_aligner(face, image):
    x, y, w, h = face.box
    if x < 0:
        x = 0
    if y < 0:
        y = 0
    if w < 0:
        w = 0
    if h < 0:
        h = 0
    img = image[int(y):int(h), int(x):int(w)]

    left_eye_y = face.lmarks[0][1]
    right_eye_y = face.lmarks[1][1]
    left_eye_x = face.lmarks[0][0]
    right_eye_x = face.lmarks[1][0]
    if left_eye_y > right_eye_y:
        A = (right_eye_x, left_eye_y)
        # Integer -1 indicates that the image will rotate in the
clockwise direction
        direction = -1
    else:
        A = (left_eye_x, right_eye_y)
        # Integer 1 indicates that image will rotate in the counter
clockwise direction
        direction = 1

    delta_x = right_eye_x - left_eye_x
    delta_y = right_eye_y - left_eye_y
    angle = np.arctan(delta_y/delta_x)
    angle = (angle * 180) / np.pi
    # Width and height of the image
    h, w = img.shape[:2]
    # Calculating a center point of the image
    # Integer division "//"" ensures that we receive whole numbers
    center = (w // 2, h // 2)
    # Defining a matrix M and calling
    # cv2.getRotationMatrix2D method
    M = cv2.getRotationMatrix2D(center, (angle), 1.0)
    # Applying the rotation to our image using the
    # cv2.warpAffine method
    rotated = cv2.warpAffine(img, M, (w, h))

    return rotated
```

همانطور که مشاهده می‌کنید، این قطعه کد این کار را با استخراج نقاط عطف چهره و استفاده از دو نقطه‌ی عطف مشخص‌کننده مردمک چشم‌ها و تراز بر مبنای شیب خط واصل مردمک چشم‌ها انجام می‌دهد. در شکل ۳۰ می‌توانید نمونه عملکرد این تابع را مشاهده کنید.

شکل ۳۰ نمونه عملکرد تابع تراز کننده چهره

۵-۴- بررسی نتایج الگوریتم پیشنهادی

در این بخش به بررسی روند آموزش و بررسی نتایج الگوریتم‌ها و سیستم پیشنهادی در آزمایش‌های مختلف می‌پردازیم.

۱-۵-۴- معرفی آزمایش‌های انجام شده

در این بخش سعی داریم تا به معرفی آزمایش‌های انجام شده در این پژوهش بپردازیم. در این پژوهش برای بررسی تاثیر حجم داده‌ها و ویدیوها و گذر زمان در دیدن مقدار بیشتر از ویدیو بر افزایش دقت اولیه در تخمین دسته‌ی شخصیت‌ها و افراد چند آزمایش انجام شد که به ترتیب

- استفاده از ۱ دقیقه از ویدیو
- استفاده از ۱۰ دقیقه از ویدیو با ۵۰ مرحله آموزش با قابل آموزش گذاشتن وزن‌های ۲۲ لایه با ضریب یادگیری ۰.۰۰۰۰۰۳
- استفاده از ۱۰ دقیقه از ویدیو با ۵۰ مرحله آموزش با قابل آموزش گذاشتن وزن‌های ۲۲ لایه با

ضریب یادگیری ۰.۰۰۰۱

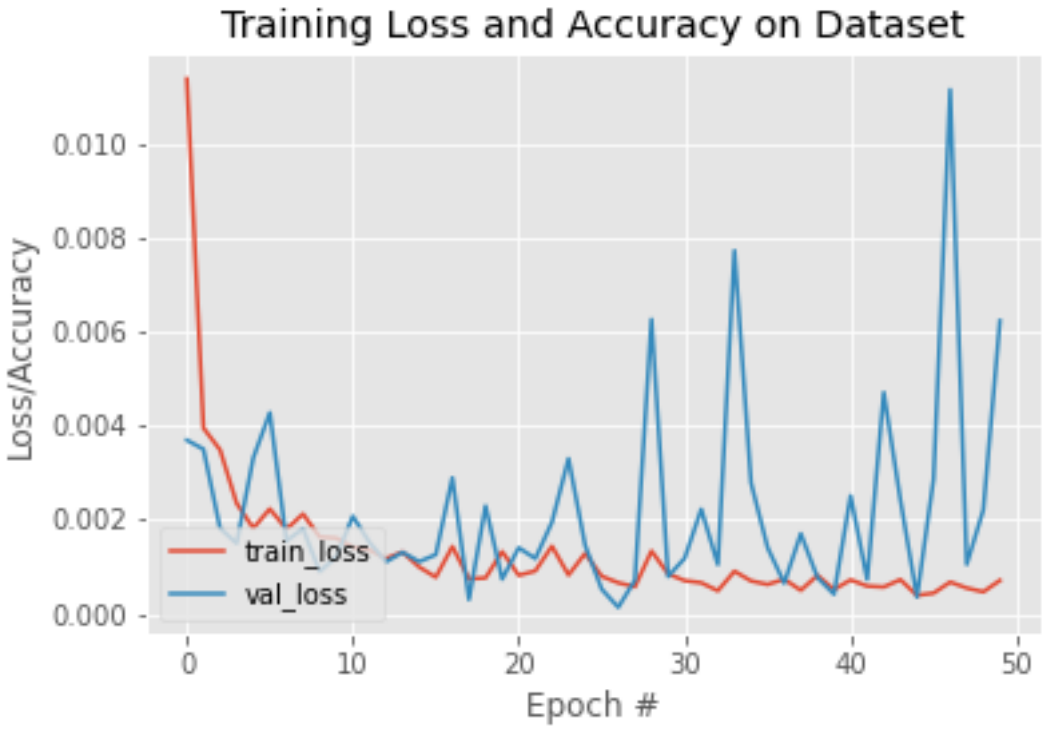
- استفاده از ۱۰ دقیقه از ویدیو با ۱۰۰ مرحله آموزش با قابل آموزش گذاشتن وزن‌های ۲۲ لایه با ضریب یادگیری ۰.۰۰۰۱
- استفاده از ۱۰ دقیقه از ویدیو با ۵۰ مرحله آموزش با قابل آموزش گذاشتن وزن‌های ۲۲ لایه با ضریب یادگیری ۰.۰۰۱
- استفاده از ۱۰ دقیقه از ویدیو با ۵۰ مرحله آموزش با قابل آموزش گذاشتن وزن‌های ۶ لایه با ضریب یادگیری ۰.۰۰۰۱

می‌باشند. در ادامه به بررسی روند آموزش و کاهش ضرر و نتایج شبکه آموزش دیده روی داده‌های تست می‌پردازیم.

۲-۵-۴- بررسی روند توابع ضرر در طی فرآیند آموزش و ارزیابی آموزش

در این قسمت به بررسی روند توابع ضرر در طی فرآیند آموزش می‌پردازیم:

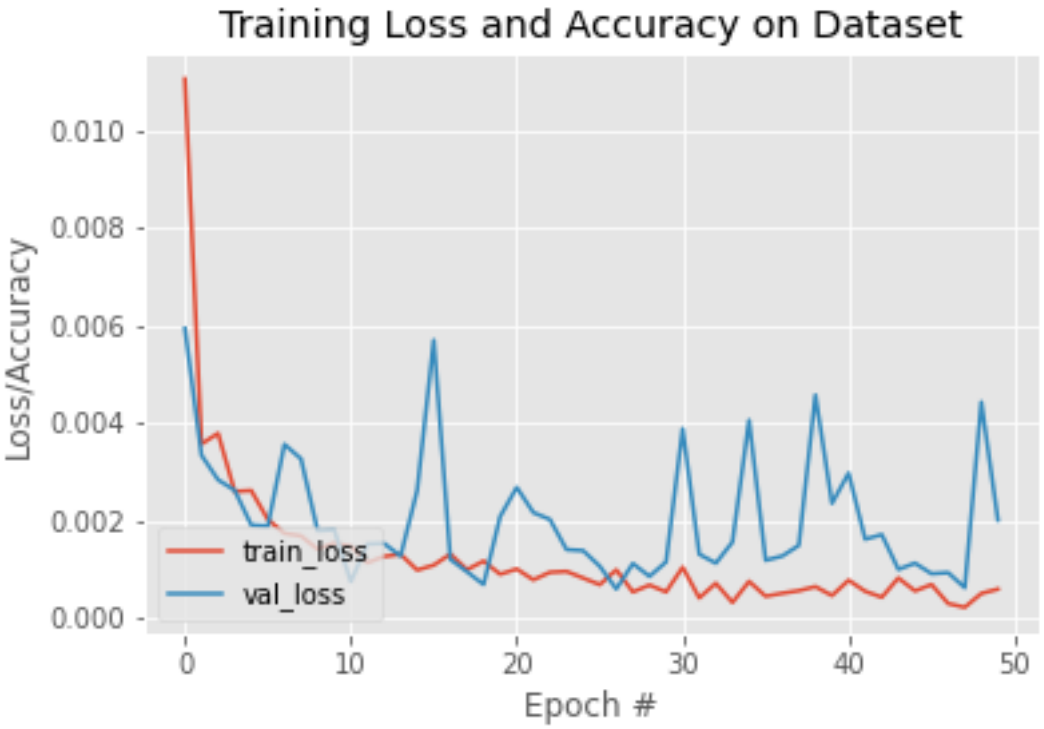
- در اولین آزمایش ۱ دقیقه از ویدیو در نظر گرفته شده است. در ۱ دقیقه از ویدیو دقت اولیه‌ی سیستم در دسته‌بندی درست کاراکترهای استخراج شده بصورت خودنظاره‌گر برابر با ۱۰۰ درصد می‌باشد و پس از تنظیم دقیق‌تر شبکه نیز دقت همان ۱۰۰ می‌باشد. و عملا متوجه این مورد می‌شویم که تاثیر طول ویدیوی دیده شده در دقت اولیه تاثیر گذار است. همچنین به دلیل ۱۰۰ درصد بودن دقت اولیه جایی برای بهبود دقت نمی‌ماند.
- در آزمایش دوم، ۱۰ دقیقه از ویدیو جهت مقایسه دقت اولیه سیستم و دقت پس از تنظیم دقیق‌تر وزن‌های شبکه عصبی در نظر گرفته شده است و ۲۴ دقیقه از ویدیو جهت تنظیم دقیق‌تر وزن‌های شبکه عصبی و ایجاد سیگناهای مثبت و منفی در نظر گرفته شده است. مشاهده می‌شود که با بررسی حجم بیشتر از ویدیو دقت اولیه‌ی سیستم در دسته‌بندی درست کاراکترهای استخراج شده بصورت خودنظاره‌گر تا ۹۷.۶۷ درصد پایین می‌آید پس هرچه بیشتر سیستم از ویدیو مشاهده کند جا برای یادگیری بیشتر و بهبود دقت بیشتر می‌شود. همچنین نویز بیشتری در داده‌ها در قالب استخراج تصاویری جز چهره مشاهده می‌شود که بتوان رفع کرد و در دسته‌ی نویز قرار داد. در نمودار ۱ روند کاهشی ضرر طی فرآیند آموزش و اعتبار سنجی آن در هر مرحله نمایش داده شده است. در نهایت فرآیند یادگیری بعد از ۵۰ مرحله آموزش متوقف می‌شود. در این حالت مشاهده می‌کنیم که بعد از ۲۵ مرحله دقت ارزیابی کاهش یافته و overfit رخ می‌دهد. همچنین هیچ‌کدام از موارد دقت یادگیری و دقت ارزیابی به عدد خاصی همگرا نمی‌شوند.



نمودار ۱ روند ضرر در آموزش با ۲۲ لایه قابل آموزش و ضریب یادگیری ۰.۰۰۰۰۰۳ و ۵۰ مرحله آموزش

- در آزمایش سوم، مشابه با آزمایش دوم، ۱۰ دقیقه از ویدیو جهت مقایسه دقت اولیه سیستم و دقت پس از تنظیم دقیق‌تر وزن‌های شبکه عصبی در نظر گرفته شده است و ۲۴ دقیقه از ویدیو جهت تنظیم دقیق‌تر وزن‌های شبکه عصبی و ایجاد سیگناهای مثبت و منفی در نظر گرفته شده است. همچنین مشابه آزمایش دوم، مشاهده می‌شود که با بررسی حجم بیشتر از ویدیو دقت اولیه‌ی سیستم در دسته‌بندی درست کاراکترهای استخراج شده بصورت خودنظاره‌گر تا ۹۷.۶۷ درصد پایین می‌آید پس هرچه بیشتر سیستم از ویدیو مشاهده کند جا برای یادگیری بیشتر و بهبود دقت بیشتر می‌شود. همچنین مشابه آزمایش دوم، نویز بیشتری در داده‌ها در قالب استخراج تصاویری جز چهره مشاهده می‌شود که بتوان رفع کرد و در دسته‌ی نویز قرار داد. در نمودار ۲ روند کاهشی ضرر طی فرآیند آموزش و اعتبار سنجی آن در هر مرحله نمایش داده شده است. در نهایت فرآیند یادگیری بعد از ۵۰ مرحله آموزش متوقف می‌شود. در این حالت مشاهده می‌کنیم

که با ضریب یادگیری ۰.۰۰۰۱ دقت ارزیابی کاهش نمی‌یابد اما نوسان می‌کند. همچنین با اینکه دقت ارزیابی در برخی مراحل با ضریب یادگیری جدید بهتر از حالت اولیه عمل می‌کند ولی هیچ‌کدام از موارد دقت یادگیری و دقت ارزیابی همچنان به عدد خاصی همگرا نمی‌شوند.



نمودار ۲ روند ضرر در آموزش با ۲۲ لایه قابل آموزش و ضریب یادگیری ۰.۰۰۰۱ و ۵۰ مرحله آموزش

- در آزمایش چهارم، مشابه با آزمایش دوم، ۱۰ دقیقه از ویدیو جهت مقایسه دقت اولیه سیستم و دقت پس از تنظیم دقیق‌تر وزن‌های شبکه عصبی در نظر گرفته شده است و ۲۴ دقیقه از ویدیو جهت تنظیم دقیق‌تر وزن‌های شبکه عصبی و ایجاد سیگناهای مثبت و منفی در نظر گرفته شده است. همچنین مشابه آزمایش دوم، مشاهده می‌شود که با بررسی حجم بیشتر از ویدیو دقت اولیه‌ی سیستم در دسته‌بندی درست کاراکترهای استخراج شده بصورت خودنظاره‌گر تا ۹۷.۶۷ درصد پایین می‌آید پس هرچه بیشتر سیستم از ویدیو مشاهده کند جا برای یادگیری بیشتر و بهبود دقت بیشتر می‌شود. همچنین مشابه آزمایش دوم، نویز بیشتری در داده‌ها در قالب استخراج تصاویری جز چهره مشاهده می‌شود که بتوان رفع کرد و در دسته‌ی نویز قرار داد. در نمودار ۳ روند کاهشی ضرر طی فرآیند آموزش و اعتبار سنجی آن در هر مرحله نمایش داده شده است. در

نهایت فرآیند یادگیری بعد از ۱۰۰ مرحله آموزش متوقف می‌شود. در این حالت مشاهده می‌کنیم که با ضریب یادگیری ۰.۰۰۰۱ بعد از ۸۰ مرحله دقت ارزیابی کاهش یافته و overfit رخ می‌دهد. همچنین با اینکه دقت ارزیابی با ضریب یادگیری جدید بهتر از حالت اولیه عمل می‌کند ولی هیچ‌کدام از موارد دقت یادگیری و دقت ارزیابی همچنان به عدد خاصی همگرا نمی‌شوند.

نمودار ۳ روند ضرر در آموزش با ۲۲ لایه قابل آموزش و ضریب یادگیری ۰.۰۰۰۱ و ۱۰۰ مرحله آموزش

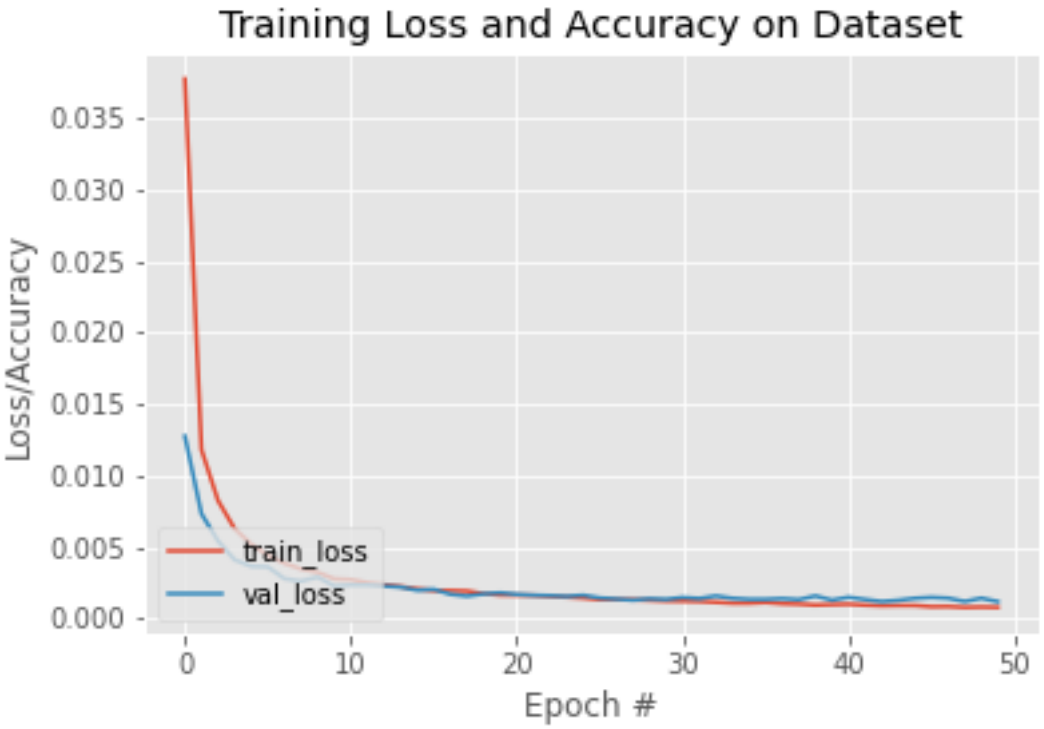
- در آزمایش پنجم، مشابه با آزمایش دوم، ۱۰ دقیقه از ویدیو جهت مقایسه دقت اولیه سیستم و دقت پس از تنظیم دقیق‌تر وزن‌های شبکه عصبی در نظر گرفته شده است و ۲۴ دقیقه از ویدیو جهت تنظیم دقیق‌تر وزن‌های شبکه عصبی و ایجاد سیگناهای مثبت و منفی در نظر گرفته شده است. همچنین مشابه آزمایش دوم، مشاهده می‌شود که با بررسی حجم بیشتر از ویدیو دقت اولیه‌ی سیستم در دسته‌بندی درست کاراکترهای استخراج شده بصورت خودنظاره‌گر تا ۹۷.۶۷ درصد پایین می‌آید پس هرچه بیشتر سیستم از ویدیو مشاهده کند جا برای یادگیری بیشتر و بهبود دقت بیشتر می‌شود. همچنین مشابه آزمایش دوم، نویز بیشتری در داده‌ها در قالب استخراج تصاویری جز چهره مشاهده می‌شود که بتوان رفع کرد و در دسته‌ی نویز قرار داد. در نمودار ۴

روند کاهشی ضرر طی فرآیند آموزش و اعتبار سنجی آن در هر مرحله نمایش داده شده است. در نهایت فرآیند یادگیری بعد از ۵۰ مرحله آموزش متوقف می‌شود. در این حالت مشاهده می‌کنیم که با ضریب یادگیری ۰.۰۰۱ دقت ارزیابی کاهش نمی‌یابد و عملا رفتاری کاملا تصادفی نشان می‌دهد. همچنین هیچ‌کدام از موارد دقت یادگیری و دقت ارزیابی همچنان به عدد خاصی همگرا نمی‌شوند.

نمودار ۴ روند ضرر در آموزش با ۲۲ لایه قابل آموزش و ضریب یادگیری ۰.۰۰۱ و ۵۰ مرحله آموزش

- در آزمایش ششم، مشابه با آزمایش دوم، ۱۰ دقیقه از ویدیو جهت مقایسه دقت اولیه سیستم و دقت پس از تنظیم دقیق‌تر وزن‌های شبکه عصبی در نظر گرفته شده است و ۲۴ دقیقه از ویدیو جهت تنظیم دقیق‌تر وزن‌های شبکه عصبی و ایجاد سیگناهای مثبت و منفی در نظر گرفته شده است. همچنین مشابه آزمایش دوم، مشاهده می‌شود که با بررسی حجم بیشتر از ویدیو دقت اولیه‌ی سیستم در دسته‌بندی درست کاراکترهای استخراج شده بصورت خودنظاره‌گر تا ۹۷.۶۷ درصد پایین می‌آید پس هرچه بیشتر سیستم از ویدیو مشاهده کند جا برای یادگیری بیشتر و بهبود دقت بیشتر می‌شود. همچنین مشابه آزمایش دوم، نویز بیشتری در داده‌ها در قالب استخراج

تصاویری جز چهره مشاهده می‌شود که بتوان رفع کرد و در دسته‌ی نویز قرار داد. امری که در این آزمایش متفاوت از آزمایش‌های قبلی است این است که در این آزمایش فقط وزن‌های ۶ لایه از شبکه عصبی را قابل آموزش می‌گذاریم. در نمودار ۵ روند کاهشی ضرر طی فرآیند آموزش و اعتبار سنجی آن در هر مرحله نمایش داده شده است. در نهایت فرآیند یادگیری بعد از ۵۰ مرحله آموزش متوقف می‌شود. در این حالت مشاهده می‌کنیم که با ضریب یادگیری ۰.۰۰۰۱ که در آزمایش‌های قبلی دریافتیم که بهینه‌ترین ضریب آموزش برای آزمایش ما می‌باشد، دقت یادگیری و دقت ارزیابی هر دو کاهش می‌یابند و به عددی یکسان همگرا می‌شوند.



نمودار ۵ روند ضرر در آموزش با ۶ لایه قابل آموزش و ضریب یادگیری ۰.۰۰۰۱ و ۵۰ مرحله آموزش

۳-۵-۴- بررسی نتایج آزمایش الگوریتم برروی داده‌ها

در این بخش به ارائه و بررسی نتایج آزمایش‌ها می‌پردازیم و دلایل رخداد آن‌ها را تحلیل می‌کنیم.

نتیجه این آزمایش‌ها در جدول ۴ نشان می‌دهد که:

- دقت اولیه‌ی سیستم بدون تنظیم دقیق‌تر وزن‌های شبکه در استخراج و دسته‌بندی چهره‌ها از

ویدیو بسیار مطلوب است و به عدد ۹۷.۶۷ درصد می‌رسد.

- هر چه طول ویدیوی مورد استفاده در فرآیند یادگیری و تنظیم دقیق‌تر شبکه بیشتر باشد، شبکه بهتر می‌تواند فضای ویدیو و شخصیت‌ها را بیاموزد و شبکه‌ی عصبی استخراج کننده بردارهای ویژگی را شکل دهد و درواقع بهبود عملکرد بیشتری می‌توان با تنظیم دقیق‌تر وزن‌های شبکه ایجاد کند.
- بدلیل اینکه وزن‌های اولیه‌ی شبکه، وزن‌های vggface می‌باشند و بسیار بهینه هستند و روی مجموعه‌داده بسیار بزرگی تنظیم شده‌اند، تنظیم دقیق‌تر آن‌ها کار ساده‌ای نیست و همان‌طور که دیدیم تنظیم مجدد و دقیق‌تر وزن‌های لایه‌های زیادی از آن موجب برهم زدن شبکه شده و دقت را کاهش می‌دهد. با توجه به اینکه سیستم ما نیز مانند vggface در همان فضای تصاویر چهره فعالیت می‌کند و مجموعه‌داده‌ی ما نیز کوچک است بهتر است که وزن‌های تعداد لایه‌های کمتری از آن را تنظیم مجدد کنیم.
- همانطور که مشاهده می‌شود، دقت اولیه‌ی سیستم نیز بسیار بالا است و بهبود بیشتر آن کار ساده‌ای نیست اما همانطور که از آزمایش‌ها مشخص است با دیدن حجم بیشتر از ویدیو دقت اولیه پایین‌تر می‌آید و خطا بیشتر رخ می‌دهد. در نتیجه جا برای رفع خطا بازتر می‌شود و سیستم بهتر می‌تواند با یادگیری برخط خود به افزایش دقت کمک کند.
- با دستیابی به تنظیمات بهینه برای تنظیم دقیق‌تر وزن‌های شبکه عصبی در آزمایش ششم ملاحظه کردیم که دقت دسته‌بندی تصاویر چهره‌ی شخصیت‌ها در ویدیوی مورد بررسی بیشتر نشد که یکی از دلایل این موضوع بالا بودن دقت اولیه است. به همین دلیل برای بررسی دقیق‌تر تغییرات شبکه عصبی شباهت کسینوسی بردارهای ویژگی استخراج شده توسط شبکه عصبی برای مجموعه‌داده مورد استفاده در فاز تنظیم دقیق‌تر وزن‌های شبکه را قبل و بعد از تنظیم دقیق‌تر وزن‌های شبکه برای هر سه‌گانه تصویر ورودی بصورت ده مقدار زیر:
    - شباهت کسینوسی بردار ویژگی تصویر پایه و زوج مثبت
    - شباهت کسینوسی بردار ویژگی تصویر پایه و زوج منفی

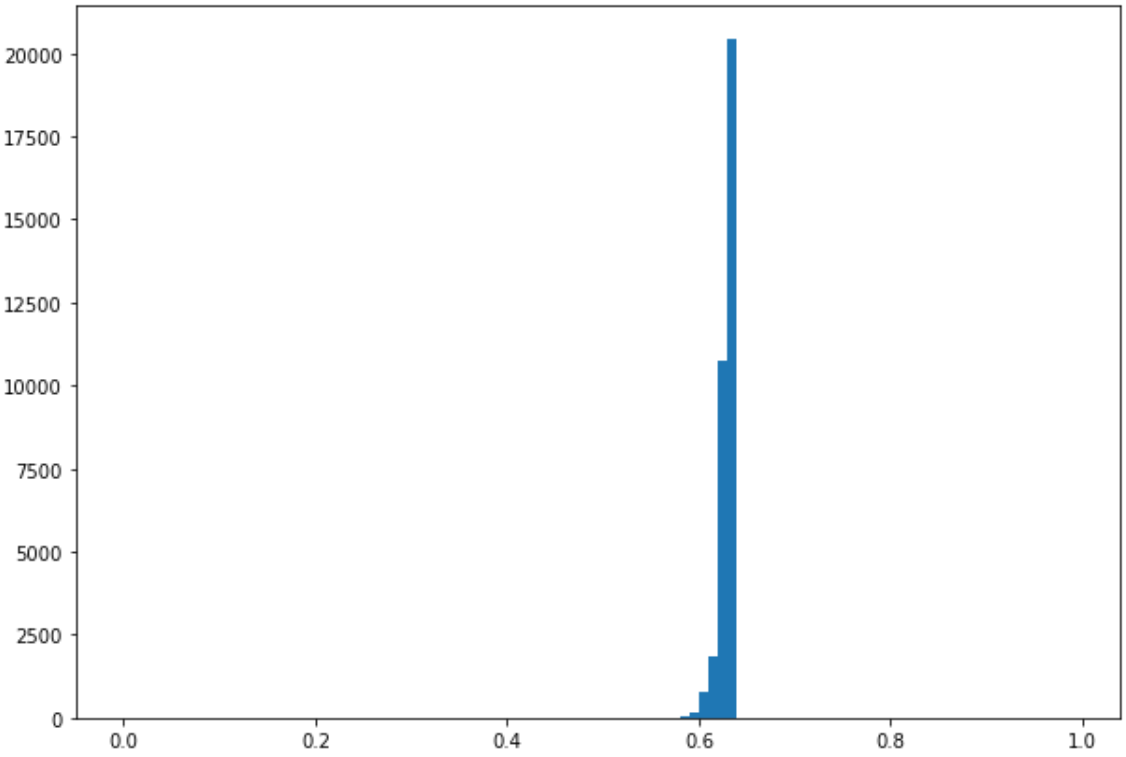
محاسبه کردیم و هیستوگرام آن‌ها را رسم کردیم و همانطور که در نمودارهای ۲ تا ۵ قابل مشاهده است بازه‌ی مقادیر بازتر شده است که این خود به بهبود دقت دسته‌بند کمک می‌کند و همچنین این مقادیر، مقداری هم شیفت داده شده‌اند که در دقت دسته‌بند برای ما امری خنثی و بی‌تاثیر خواهد بود.

جدول ۴ نتایج دسته‌بندی قبل و بعد از تنظیم

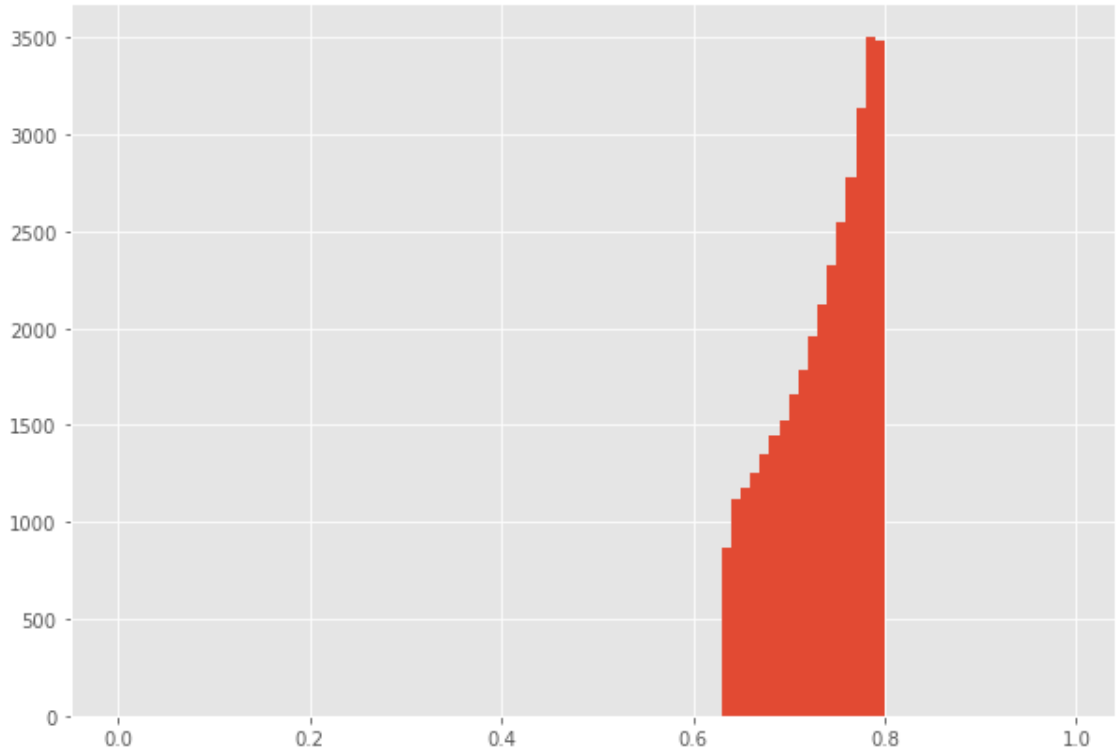
شماره آزمایش	طول ویدیو	تعداد مراحل آموزش	دقت اولیه (NMI)	دقت پس از یادگیری (NMI)
۱	۱	۱۰	۱۰۰	۱۰۰
۲	۱۰	۵۰	۹۷.۶۷	۷۶.۵۴
۳	۱۰	۵۰	۹۷.۶۷	۸۴.۷۱
۴	۱۰	۱۰۰	۹۷.۶۷	۷۴.۲۱
۵	۱۰	۵۰	۹۷.۶۷	۸۹.۷۳
۶	۱۰	۵۰	۹۷.۶۷	۹۷.۶۷

جدول ۵ میانگین مقادیر شباهت کسینوسی بردارهای ویژگی قبل و بعد از تنظیم دقیق وزن‌ها

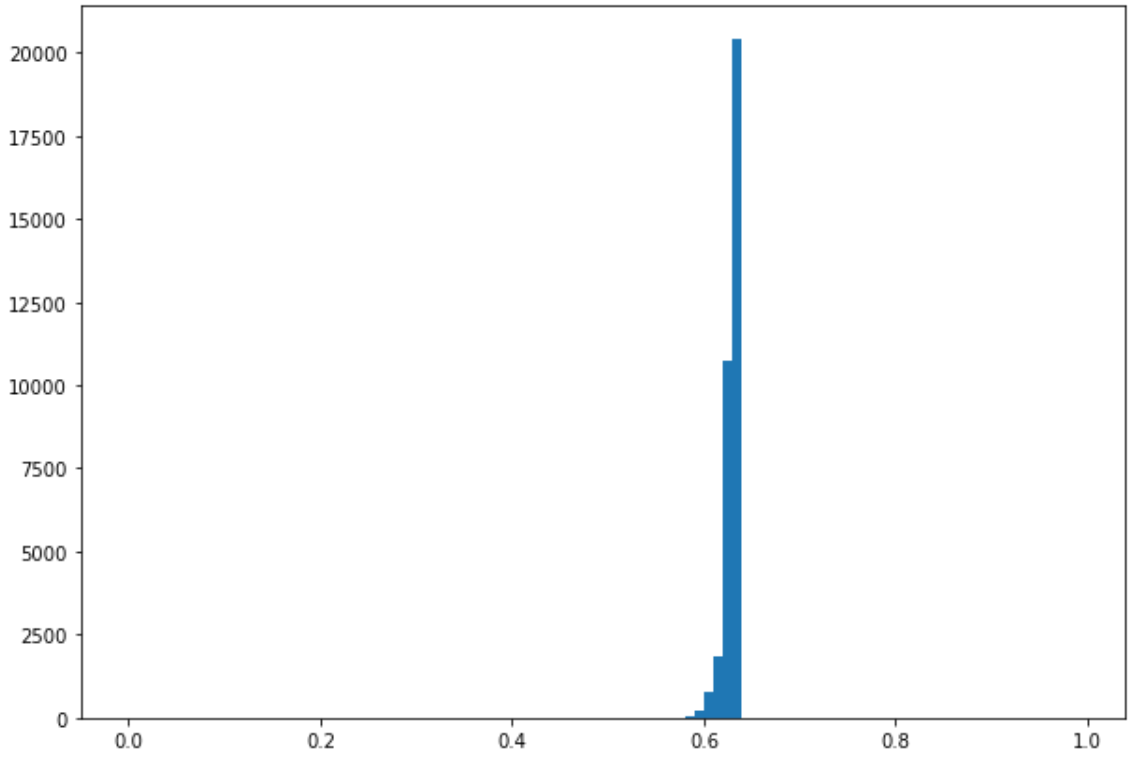
بردارها	میانگین شباهت کسینوسی قبل از آموزش	میانگین شباهت کسینوسی بعد از آموزش
تصویر پایه و زوج مثبت	۱۰.۶۶۹۹۳۳۶۷۰۶۳۹۹۹۲	۱۲.۴۸۸۰۹۶۱۷۱۲۳۰۰۷۸
تصویر پایه و زوج منفی	۱۰.۶۶۹۲۳۸۳۶۲۵۵۰۷۳۶	۱۲.۴۸۸۰۵۸۹۱۴۸۹۹۸۲۶



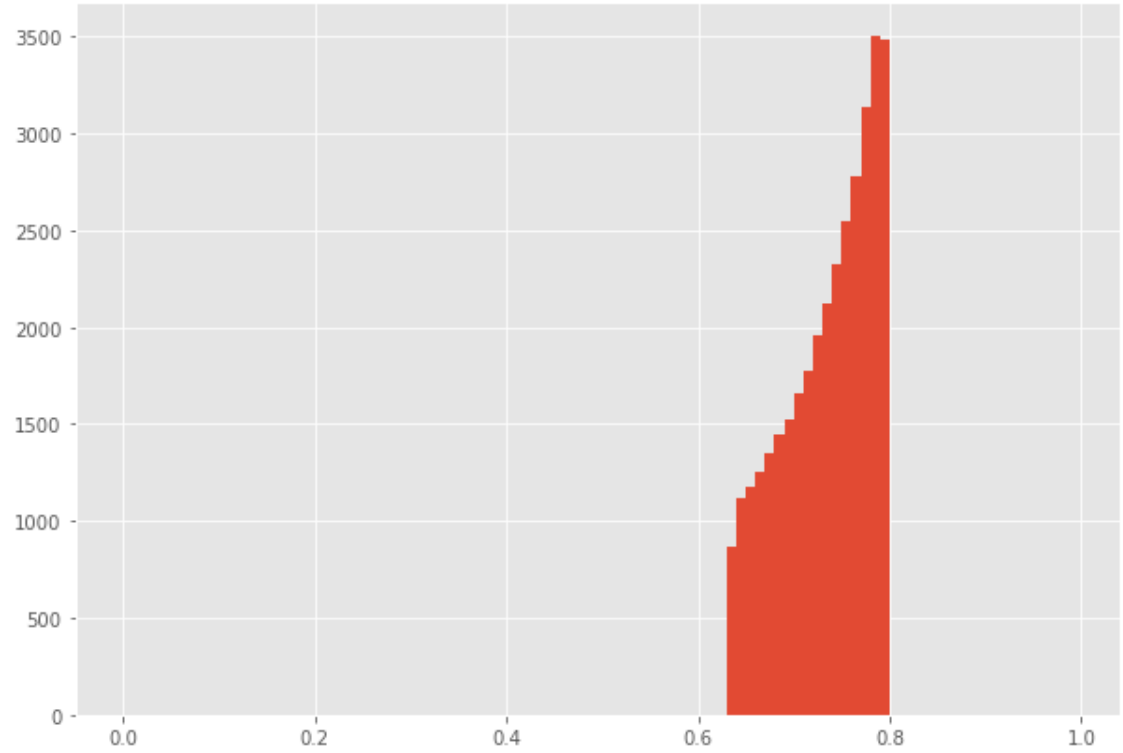
نمودار ۳ هیستوگرام شباهت کسینوسی بردار ویژگی‌های تصاویر پایه و تصاویر زوج مثبت قبل از آموزش



نمودار ۴ هیستوگرام شباهت کسینوسی بردار ویژگی‌های تصاویر پایه و تصاویر زوج مثبت بعد از آموزش



نمودار ۵ هیستوگرام شباهت کسینوسی بردار ویژگی‌های تصاویر پایه و تصاویر زوج منفی قبل از آموزش



نمودار ۶ هیستوگرام شباهت کسینوسی بردار ویژگی‌های تصاویر پایه و تصاویر زوج منفی بعد از آموزش

۶-۴- نتیجه گیری

چالش‌های اصلی در این پژوهش شامل

- آشکار سازی تصویر چهره در فریم‌های ویدیو
- یافتن ارتباط تصاویر چهره‌ی استخراج شده و یافتن دنباله‌های چهره از روی آن‌ها
- استخراج بردار ویژگی از تصاویر چهره‌های استخراج شده
- دسته‌بندی دنباله‌های تصاویر چهره‌های استخراج شده بر اساس بردارهای ویژگی میانگین آن‌ها
- پیاده‌سازی شبکه Siamese با تابع ضرر سه‌گانه جهت تنظیم دقیق‌تر وزن‌های شبکه عصبی استخراج بردارهای ویژگی

بود که در هر بخش سعی شد با ارائه راه حل‌های مناسب عملکرد الگوریتم پیشنهادی در حل مسئله را بهبود بخشید. استفاده از آشکار ساز چهره RetinaFace و شبکه عصبی با وزن‌های از پیش آموخته vggface و دسته‌بند سلسله مراتبی و شبکه Siamese با تابع ضرر سه‌گانه در فرآیند آموزش و تنظیم دقیق‌تر وزن‌های شبکه راه حل‌هایی بودند که برای فائق آمدن بر چالش‌های موجود در زمینه‌ی طراحی سیستم‌های بررسی و آنالیز خودکار ویدیو و آموزش برخط آن‌ها در این پژوهش در نظر گرفته‌شد و نتایج آن مورد بررسی قرار گرفت.

۷-۴- کارهای آینده

از جمله کارهایی که می‌توان در راستای این پژوهش در ادامه انجام داد می‌توان به موارد زیر اشاره کرد:

- پیاده‌سازی تابع ضرر بهینه‌تر در تنظیم دقیق‌تر وزن‌های شبکه
- استفاده‌ی بهینه از نشانه‌های[1] چهره در دسته‌بندی و تنظیم دقیق‌تر وزن‌های شبکه

[1] Landmark

مراجع

[۱] M. Tapaswi, M. T. Law و S. Fidler, “Video Face Clustering With Unknown Number of Clusters ”,*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* ,۲۰۱۹ .

[۲] V. Sharma, M. Tapaswi, M. S. Sarfraz و Rainer Stiefelhagen, “Video Face Clustering With Self-Supervised Representation Learning ”,*IEEE Transactions on Biometrics, Behavior, and Identity Science* , جلد ۲, شماره ۲, pp. ۱۴۵-۱۵۷, ۲۰۲۰ .

[۳] K. Zhang, Z. Zhang, Z. Li و Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks ”,*IEEE Signal Processing Letters* ,۱۰ شماره ,۲۳ جلد ,pp. ۱۴۹۹-۱۵۰۳, ۲۰۱۶ .

[۴] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu و A. C. Berg, “SSD: Single Shot MultiBox Detector ”,*ECCV* .۲۰۱۶ ,۹۹۰۵ جلد ,

[۵] G. Castellano و G. Vessio, “A Deep Learning Approach to Clustering Visual Arts ”,*International Journal of Computer Vision* ,۱۳۰ جلد ,pp. ۲۵۹۰-۲۶۰۵, ۲۰۲۲ .

[۶] “شبکه عصبی مصنوعی,” [درون خطی]. Available: https://fa.wikipedia.org/wiki/شبکه_عصبی_مصنوعی. [دستیابی در ۲۰۲۰].

[۷] “کاربرد دیتاست در Data Mining”. [درون خطی]. Available: https://nikamooz.com/dataset-datamining/. [دستیابی در ۲۰۲۰]

[۸] Girshick, Ross and Donahue, Jeff and Darrell, Trevor and Malik, Jitendra, “Region-based convolutional networks for accurate object detection and segmentation ”,*IEEE transactions on pattern analysis and machine intelligence* ,۳۸ جلد ,pp. ۱۴۲--۱۵۸, ۲۰۱۵ .

[۹] Girshick, Ross, “Fast r-cnn در ”,*Proceedings of the IEEE international conference on computer vision* ,۲۰۱۵ ,pp. ۱۴۴۰--۱۴۴۸.

[۱۰] Ren, Shaoqing and He, Kaiming and Girshick, Ross and Sun, Jian, “Faster r-cnn: Towards real-time object detection with region proposal networks ”,*arXiv preprint arXiv:۱۵۰۶.۰۱۴۹۷* .۲۰۱۵ ,

[۱۱] Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali, “You only look once: Unified, real-time object detection در ”,

Proceedings of the IEEE conference on computer vision and pattern recognitio ,۲۰۱۶ ,pp. ۷۷۹--۷۸۸.

[۱۲] Liu, Wei and Anguelov, Dragomir and Erhan, Dumitru and Szegedy, Christian and Reed, Scott and Fu, Cheng-Yang and Berg, Alexander C, "Ssd: Single shot multibox detector," in *European conference on computer vision*, Springer, ۲۰۱۶, pp. ۲۱--۳۷.

[۱۳] J. Deng, J. Guo, E. Ververas, I. Kotsia و S. Zafeiriou, "RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild ",*IEEE/CVF Conference on Computer Vision and Pattern Recognition* ,pp. ۵۲۰۲-۵۲۱۱, ۲۰۲۰ .

[۱۴] Lin, Tsung-Yi and Goyal, Priya and Girshick, Ross and He, Kaiming and Doll'ar, Piotr, "Focal loss for dense object detection در ",*Proceedings of the IEEE international conference on computer vision* ,۲۰۱۷ ,pp. ۲۹۸۰--۲۹۸۸.

[۱۵] "Video Tracking .[درون خطی] ",Available: https://en.wikipedia.org/wiki/Video_tracking [دستیابی در July ۲۰۲۱.]

[۱۶] "Cluster Analysis .[درون خطی] ",Available: https://en.wikipedia.org/wiki/Cluster_analysis.[دستیابی در ۲۰۲۱] .

[۱۷] A. Rosebrock, ۲۳ July ۲۰۱۸ .[درون خطی] .Available: https://pyimagesearch.com/۲۰۱۸/۰۷/۲۳/simple-object-tracking-with-opencv.[دستیابی در ۲۰۲۱] /.

[۱۸] "خوشه‌بندی سلسله مراتبی," [درون خطی]. .Available: https://fa.wikipedia.org/wiki/%D۸%AE%D۹%۸۸%D۸%B۴%D۹%۸۷%E۲%۸۰%۸C%D۸%A۸%D۹%۸۶%D۸%AF%DB%۸C_%D۸%B۳%D۹%۸۴%D۸%B۳%D۹%۸۴%D۹%۸۷%E۲%۸۰%۸C%D۹%۸۵%D۸%B۱%D۸%A۷%D۸%AA%D۸%A۸%DB%۸C.[دستیابی در ۲۰۲۲] .

[۱۹] "Siamese neural network .[درون خطی] ",Available: https://en.wikipedia.org/wiki/Siamese_neural_network.[دستیابی در ۲۰۲۲] .

[۲۰] H. Essam و S. L. Valdarrama, ۲۵ March ۲۰۲۱ .[درون خطی] .Available: https://keras.io/examples/vision/siamese_network.[دستیابی در ۲۰۲۲] /.

[۲۱] A. Rosebrock, "Intersection over Union (IoU) for object detection," ۷ November ۲۰۱۶ .[درون خطی] .Available:

https://pyimagesearch.com/۲۰۱۶/۱۱/۰۷/intersection-over-union-iou-for-object-detection.[دستیابی در ۲۰۲۲]./

[۲۲] “Clustering .[درون خطی] ”,Available: https://scikit-learn.org/stable/modules/clustering.html.[دستیابی در ۲۰۲۱] .

[۲۳] “Agglomerative Clustering .[درون خطی] ”,Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.[دستیابی در ۲۰۲۲] .

[۲۴] “image-net .[درون خطی] ”,Available: http://www.image-net.org [دستیابی در ۱۸ ۲ ۲۰۲۱]./ .

[۲۵] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian, “Deep residual learning for image recognition در ”,*Proceedings of the IEEE conference on computer vision and pattern recognition* ,۲۰۱۶ ,pp. ۷۷۰--۷۷۸.

[۲۶] F. Shroff, D. Kalenichenko و J. Philbin, “FaceNet: A unified embedding for face recognition and clustering ”,*IEEE Conference on Computer Vision and Pattern Recognition* ,pp. ۸۱۵-۸۲۳, ۲۰۱۵ .

[۲۷] I. Tahmasb .[فیلم] *مهمونی*. کارگردان, Iran: Namava ,۲۰۲۲ .